RClone quickmanual one population

Diane Bailleul

2015-10-22

"Eager Beginners" Manual for RClone package

RClone data format: one population

A. Introduction to RClone

RClone is a R package version of *GenClone* program: to analyse data (SSR, SNP, . . .), test for clonality and describe spatial clonal organisation.

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 index and Edge Effect estimation.

Some of these analysis can be applied to dataset without clones.

B. RClone data format: one population

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

If you have several populations in your dataset, go to other vignette *RClone_qmsevpops*.

C. General format

If you have haploid data, you can skip to *4, For GenClone users* or *D. Description of data set.*

An *RClone* table must look like:

```r
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 simple case: you already have a one-allele per column table

```r
data(posidonia)

sort_all(posidonia)
```

2, The classic case: one locus per column

```r
#Let's create your example table:
test <- matrix("232/231", ncol = 2, nrow = 2)
colnames(test) <- paste("locus", 1:2, sep = "_")


#Use :
data1 <- convert_GC(as.data.frame(test), 3, "/")
```

```r
data1
```

locus_1_1	locus_1_2	locus_2_1	locus_2_2
231	232	231	232
231	232	231	232

We used “3” because this is the length of the allele (with 3 numbers).
For allele separation, we used “/” because, of course, it was the separator.

3, You already work with Adegenet

It’s a kind of like the case number 2, but you have to export your `genind` data into table first:

```r
#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"
```

4, For GenClone users

Warning: your infile file must include all the informations available, as locus names and ploidy level (which is not mandatory for *GenClone*).

```r
data(infile)

#This is nearly a GenClone file, type:
write.table(infile, "infile.csv", col.names = FALSE, row.names = FALSE, sep = ";")

#Now you have a formatted GenClone file:
res <- transcript_GC("infile.csv", ";", 2, 7, 3)
posidonia <- res$data_genet
coord_posidonia <- res$data_coord
```

You might need to edit your “infile.txt” into “infile.csv” and check if there’s “.” and not “,” for geographic coordinates, and use “;” as separator element.

- “2” is for the ploidy level; should have been “1” for haploid data;
- “7” here is the number of loci;
- “3” is for allele length. Posidonia alleles are always of length “3”.

D. Description of data set

D.1 Discrimination of MLG

List unique alleles per locus:

Basic commands:

```
data(posidonia)

list_all_tab(posidonia)
```

or, for haploid data:

```
list_all_tab(haplodata, haploid = TRUE)
```

Results:

```
list_all_tab(posidonia)
```

locus_1	locus_2	locus_3	locus_4	locus_5	locus_6	locus_7
137	182	212	234	165	170	178
139	174	222	242	159	168	180
161	188	210	236	163	172	
151	180	208				
157		216				
159		226				
171		218				

List MLG:

Basic commands:

```
MLG_tab(posidonia)
```

or, for haploid data:

```
MLG_tab(haplodata)
```

Results:

```
MLG_tab(posidonia)
```

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

Allelic frequencies:

Basic commands:

```
freq_RR(posidonia)
```

or, for haploid data:

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

Options:

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

Results:

```
freq_RR(posidonia)
```

locus	allele	freq_ramet	freq_genet	freq_RR
locus_1	137	0.1375	0.1607143	0.1666667
locus_1	139	0.0250	0.0357143	0.0370370
locus_1	151	0.1500	0.2142857	0.2222222
locus_1	157	0.3375	0.2857143	0.2777778
locus_1	159	0.0250	0.0357143	0.0370370
locus_1	161	0.3125	0.2500000	0.2407407
locus_1	171	0.0125	0.0178571	0.0185185

D.2 Test for reliability of data

On loci

Basic commands:

```
sample_loci(posidonia, nbrepeat = 1000)
```

or, for haploid data:

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

Options:

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

Results:

```
res <- sample_loci(posidonia, nbrepeat = 1000, He = TRUE) #time consuming
names(res)
```

```
> NULL
```

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

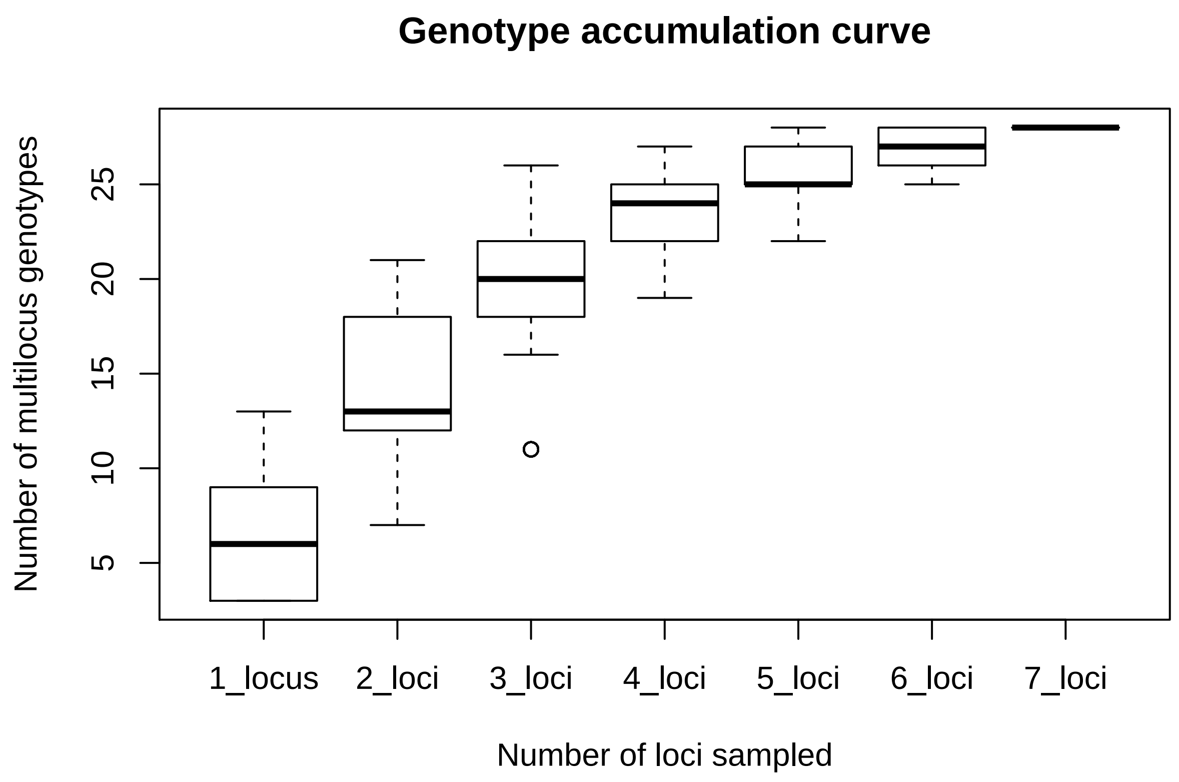
nb_loci	min	max	mean_MLG	SE
1	3	13	6.265	0.1046505
2	7	21	14.265	0.1362400
3	11	26	20.142	0.0966083
4	19	27	23.566	0.0617532
5	22	28	25.443	0.0460312
6	25	28	26.856	0.0311164
7	28	28	28.000	0.0000000

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

nb_loci	min	max	mean_all	SE	He	SE
1	2	7	4.092	NA	0.5491902	NA
2	5	14	8.329	132.25780	0.5492449	1.2174962
3	8	18	12.416	88.28636	0.5503377	0.8028116
4	11	21	16.531	70.20927	0.5504794	0.6456283
5	15	24	20.699	60.66198	0.5504022	0.5523189
6	22	27	24.895	54.60655	0.5521684	0.4933410
7	29	29	29.000	NA	0.5513110	NA

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

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



Same on units

Basic commands:

```r
sample_units(posidonia, nbrepeat = 1000)
```

or, for haploid data:

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

E Determination of MLL

E.1 psex/psex Fis with pvalue computation

pgen, psex and p-values

Basic commands:

```r
pgen(posidonia)
data(factoR) #for psex
psex(posidonia)
```

or, for haploid data:

```r
pgen(haplodata, haploid = TRUE)
data(factoR) #for psex
psex(haplodata, haploid = TRUE)
```

Options: *(idem on psex and pgen)*

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

Results:

```r
data(factoR)
res <- psex(posidonia, RR = TRUE, nbrepeat = 1000)
res[[1]] #if nbrepeat != 0, res contains a table of psex values
                                  #and a vector of sim-psex values
```

pgen	genet	psex	pvalue
2.20e-06			
0.00e+00			
4.77e-05			
4.77e-05	3	0.00190284159898287	0.392857142857143
4.77e-05	3	1.76851132496336e-06	0
4.77e-05	3	1.06767920426143e-09	0

```r
res[[2]] #sim psex values
```

```
>  [1] 2.682915e-03 1.351209e-03 3.404466e-03 1.543552e-03 4.299086e-03
>  [6] 6.265958e-03 9.866499e-03 1.920650e-03 2.045403e-03 5.527621e-04
> [11] 6.364326e-04 1.374158e-03 5.837434e-03 3.624390e-03 2.895358e-03
> [16] 5.969326e-03 9.347855e-04 7.666523e-04 6.671097e-05 2.522795e-03
> [21] 5.676186e-03 1.297853e-03 1.105800e-03 5.573546e-03 2.807860e-03
> [26] 4.025514e-03 1.851704e-03 5.309521e-03
```

Fis, pgen Fis, psex Fis and p-values

Not for haploid data !

Fis

Basic commands:

```
Fis(posidonia)
```

Options:

```
Fis(posidonia) #Fis on ramets
Fis(posidonia, genet = TRUE) #Fis on genets
Fis(posidonia, 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(posidonia, RR = TRUE)[[2]]
```

locus	Hobs	Hatt	Fis
locus_1	0.6666667	0.7994410	0.1660839
locus_2	0.5185185	0.5024949	-0.0318882
locus_3	0.8846154	0.8099548	-0.0921788
locus_4	0.2962963	0.2620545	-0.1306667
locus_5	0.3214286	0.5512987	0.4169611
locus_6	0.6400000	0.6555102	0.0236613
locus_7	0.3571429	0.3818182	0.0646259

pgen Fis, psex Fis and p-values

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

```
pgen_Fis(posidonia)
```

Options:

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

Results:

```
data(factoR)
res <- psex_Fis(posidonia, RR = TRUE, nbrepeat = 1000)
res[[1]]
#if nbrepeat != 0, res contains a table of psex values
                           #and a vector of sim-psex Fis values
```

pgenFis	genet	psexFis	pvalue
1.05e-05			
0.00e+00			
4.39e-05			
4.39e-05	3	0.00175402908240928	0.258064516129032
4.39e-05	3	1.50248895374508e-06	0
4.39e-05	3	8.36013934496707e-10	0

```
res[[2]] #sim psex Fis values
```

```
>  [1] 0.0040481045 0.0031602068 0.0092107387 0.0005867821 0.0078841578
>  [6] 0.0016065540 0.0008205260 0.0037157779 0.0069945737 0.0013747738
> [11] 0.0025227684 0.0012591533 0.0131772838 0.0011010652 0.0016714224
> [16] 0.0036430883 0.0043642467 0.0009267953 0.0146375958 0.0097961140
> [21] 0.0056357471 0.0049308171 0.0105839008 0.0018554896 0.0057345994
> [26] 0.0180426243 0.0025966226 0.0045779356 0.0036178632 0.0088153811
> [31] 0.0076859203
```

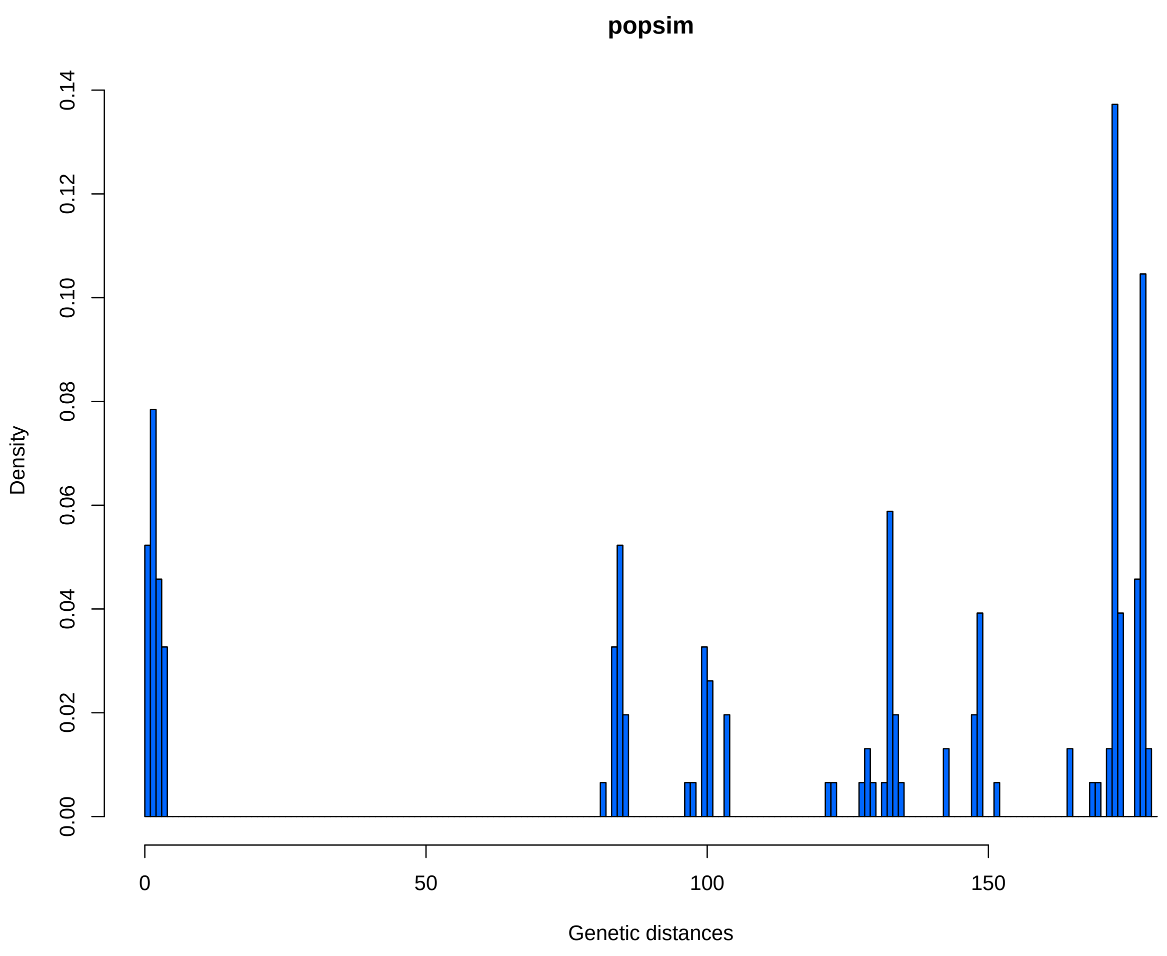
E.2 MultiLocus Lineages

Genetic distance matrix computation and threshold definition

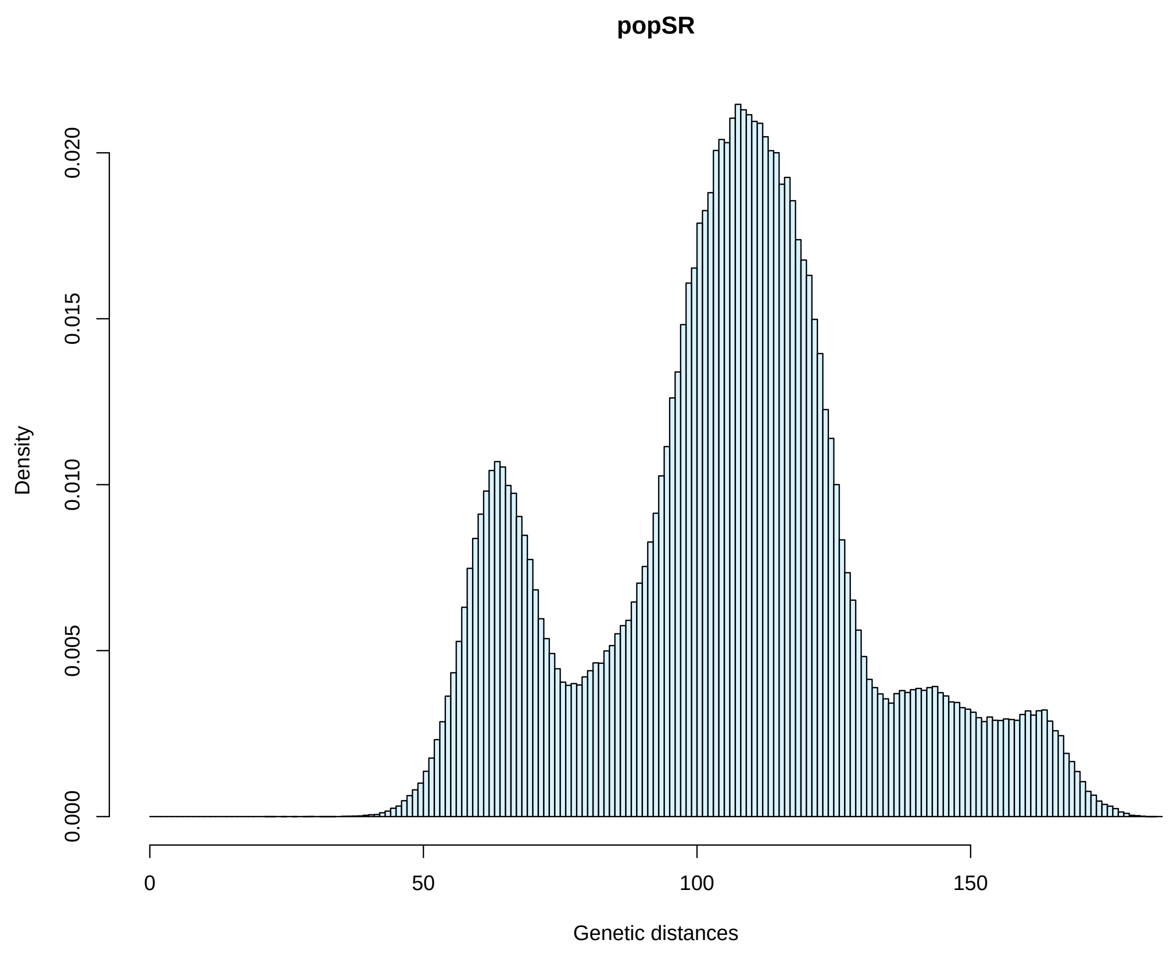
On a theoretical diploid population with c = 0.9999 (c, clonality rate).

```
data(popsim)

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

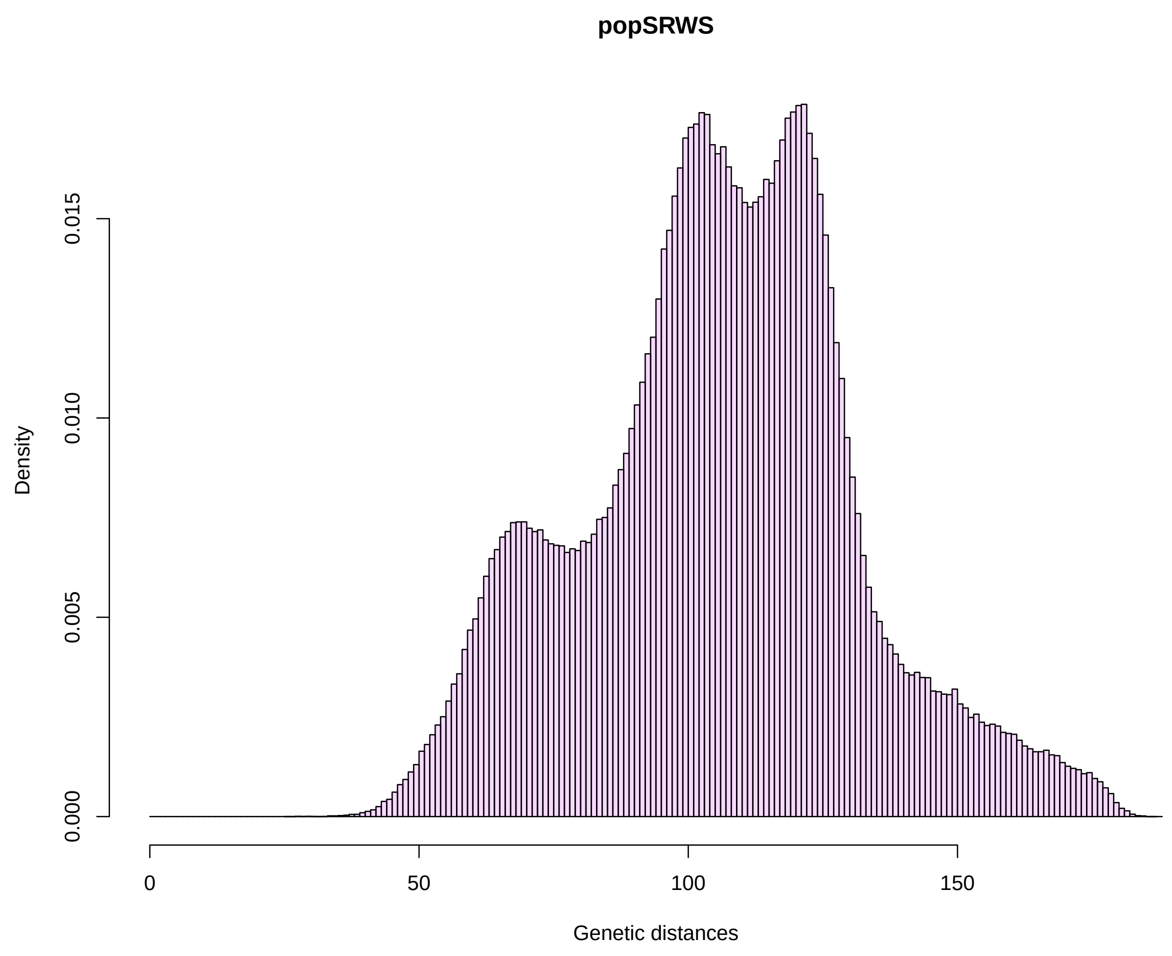
```
#graph prep.:
p1 <- hist(respop$distance_matrix, freq = FALSE, col = rgb(0,0.4,1,1), main = "popsim",
           xlab = "Genetic distances", breaks = seq(0, max(respop$distance_matrix)+1, 1))
```



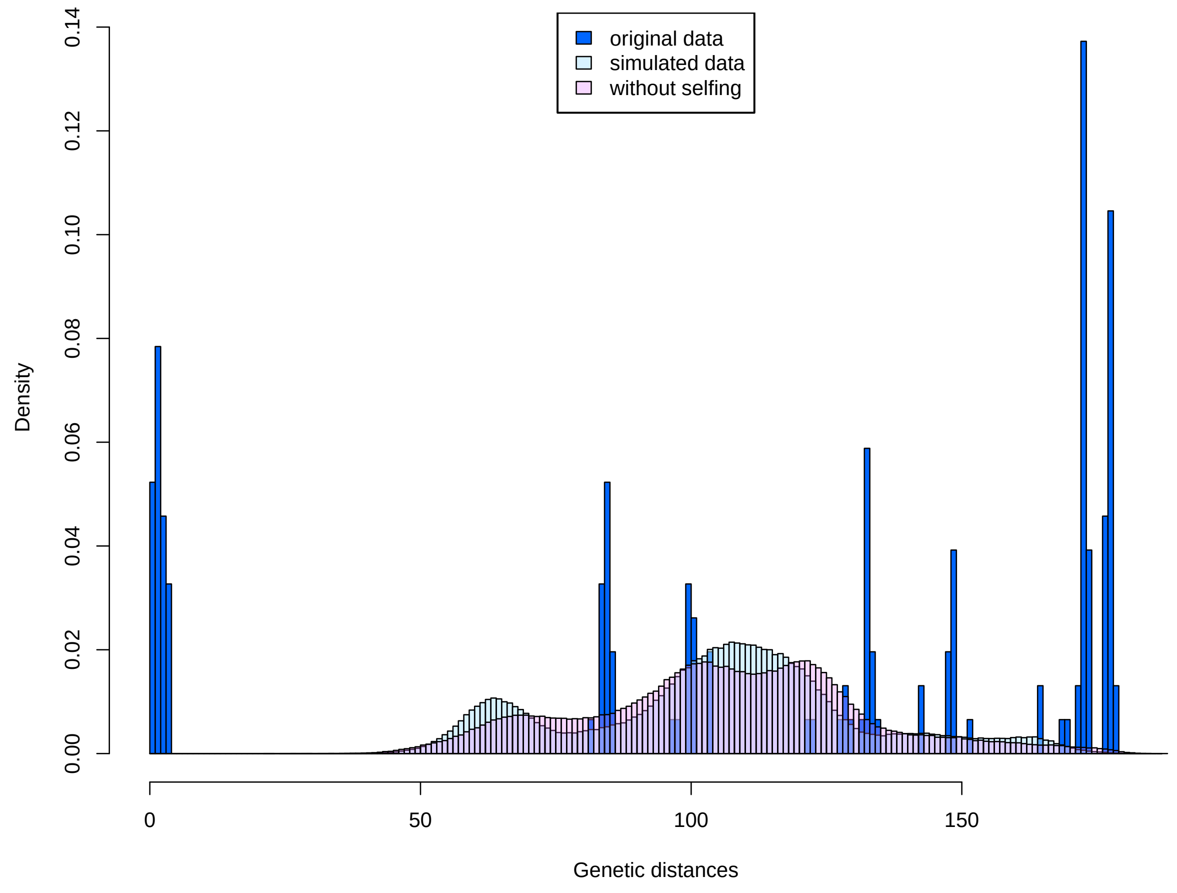
```
p2 <- hist(ressim$distance_matrix, freq = FALSE, col = rgb(0.7,0.9,1,0.5), main = "popSR",
             xlab = "Genetic distances", breaks = seq(0, max(ressim$distance_matrix)+1, 1))
```



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



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

#graph superposition:
plot(p1, col = rgb(0,0.4,1,1), freq = FALSE, xlim = c(0,limx), main = "",
        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$distance_matrix)
>
>   1   2   3   4  82  84  85  86  97  98 100 101 104 122 123 128 129 130
>   8  12   7   5   1   5   8   3   1   1   5   4   3   1   1   1   2   1
> 132 133 134 135 143 148 149 152 165 169 170 172 173 174 177 178 179
>   1   9   3   1   2   3   6   1   2   1   1   2  21   6   7  16   2
#alpha2 = 4
```

```
#creating MLL list:
MLLlist <- MLL_generator(popsim, alpha2 = 4)
#or
res <- genet_dist(popsim, alpha2 = 4)
MLLlist <- MLL_generator2(res$potential_clones, MLG_list(popsim))
```

For haploid data, theoretical example:

```
respop <- genet_dist(haplodata, haploid = TRUE)
ressim <- genet_dist_sim(haplodata, haploid = TRUE, nbrepeat = 1000)
MLLlist <- MLL_generator(haplodata, haploid = TRUE, alpha2 = 4)
#or
```

```r
res <- genet_dist(haplodata, haploid = TRUE, alpha2 = 4)
MLLlist <- MLL_generator2(res$potential_clones, haploid = TRUE, MLG_list(haplodata))
```

F. Genotypic diversity and evenness indices calculation

F.1 Classic genotypic indices

Basic commands:

```r
clonal_index(posidonia)
```

or, with MLL:

```r
clonal_index(popsim, listMLL = MLLlist)
```

or, for haploid data:

```r
clonal_index(haplodata)
```

Results:

```r
clonal_index(posidonia)
```

	G	R	H"	J'	D	V	Hill
MLG	28	0.6923077	3.149621	0.9452064	0.9705128	0.7921811	33.91304

F.2 Pareto index

Basic commands:

```r
Pareto_index(posidonia)
```

or, with MLL:

```r
Pareto_index(popsim, listMLL = MLLlist)
```

or, for haploid data:

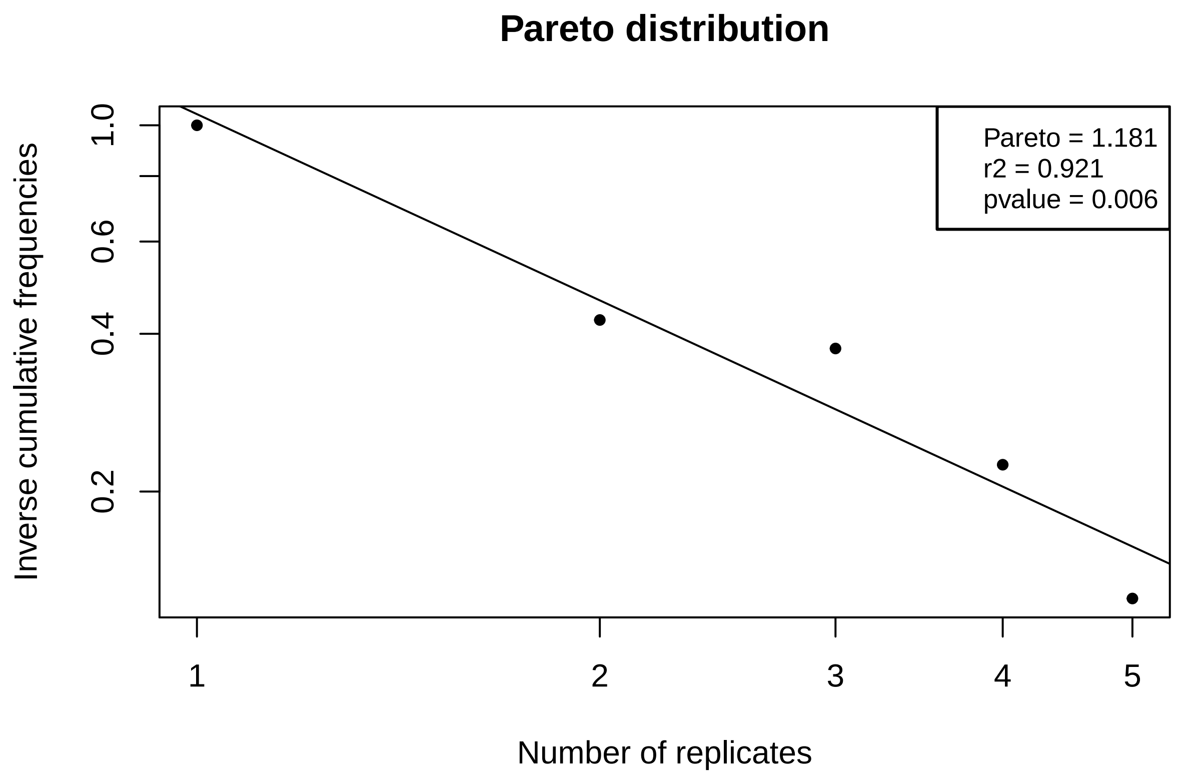
```r
Pareto_index(haplodata)
```

Options:

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

Results:

```
res <- Pareto_index(posidonia, full = TRUE, graph = TRUE, legends = 2)
```



```
names(res)
> [1] "Pareto"             "c_Pareto"            "regression_results"
> [4] "coords_Pareto"
res$Pareto
> [1] 1.180756
res$c_Pareto
> [1] 2.180756
#res$regression_results
#res$coords_Pareto #points coordinates
```

G. Spatial description of clonality

G.1 Spatial autocorrelation

Basic commands:

```r
autocorrelation(posidonia, coords = coord_posidonia, Loiselle = TRUE)
```

or, with MLL:

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

or, for haploid data:

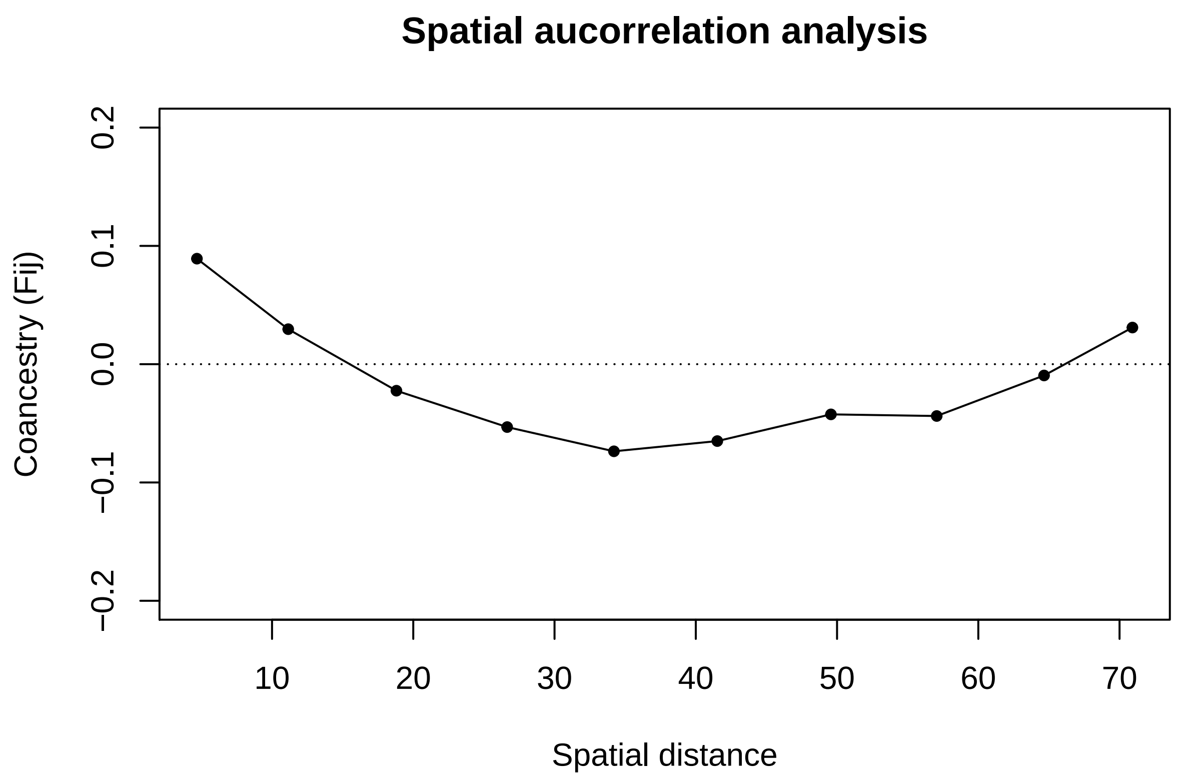
```r
autocorrelation(haplodata, haploid = TRUE, coords = coord_haplo, Loiselle = TRUE)
```

Lot's of options:

```r
data(posidonia)
data(coord_posidonia)

#kinship distances:
autocorrelation(posidonia, coords = coord_posidonia, Loiselle = TRUE)
autocorrelation(posidonia, coords = coord_posidonia, Ritland = TRUE)

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

#distance classes construction:
autocorrelation(posidonia, coords = coord_posidonia, 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(posidonia, coords = coord_posidonia, Loiselle = TRUE,
                    vecdist = distvec) #custom distance vector
autocorrelation(posidonia, coords = coord_posidonia, Loiselle = TRUE,
                    class1 = TRUE, d = 7) #7 equidistant classes
autocorrelation(posidonia, coords = coord_posidonia, Loiselle = TRUE,
                    class2 = TRUE, d = 7)
                    #7 distance classes with the same number of units in each

#graph options:
autocorrelation(posidonia, coords = coord_posidonia, Ritland = TRUE, graph = TRUE)
                                                                  #displays graph
autocorrelation(posidonia, coords = coord_posidonia, Ritland = TRUE, export = TRUE)
                                                                  #export graph

#pvalues computation
autocorrelation(posidonia, coords = coord_posidonia, Ritland = TRUE, nbrepeat = 1000)
```

Results:

```r
res <- autocorrelation(posidonia, coords = coord_posidonia, Ritland = TRUE,
                       nbrepeat = 1000, graph = TRUE)
```



```r
names(res)
```

```
> [1] "Main_results"            "Slope_and_Sp_index"
> [3] "Slope_resample"          "Kinship_resample"
> [5] "Matrix_kinship_results" "Class_kinship_results"
> [7] "Class_distance_results"
```

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

dist_min	dist_max	dist_mean	ln(dist_mean)	nb_pairs	mean_Ritland	pval_kin
0.50000	7.51665	4.683712	1.544091	97	0.0891802	0.000
7.61577	15.20691	11.148114	2.411270	157	0.0296031	0.000
15.23975	22.80351	18.807914	2.934278	119	-0.0224115	0.390
22.94014	30.41381	26.648255	3.282724	110	-0.0531668	0.000
30.50000	38.00329	34.206496	3.532416	121	-0.0736379	0.000
38.02959	45.59879	41.524146	3.726275	64	-0.0650049	0.000
46.09772	53.08484	49.568560	3.903357	34	-0.0424233	0.144
53.53737	60.66144	57.055830	4.044030	29	-0.0438132	0.154
61.00205	68.00184	64.657149	4.169099	31	-0.0095349	0.800
68.52919	76.04111	70.912179	4.261442	18	0.0309692	0.106

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

```
> mean_Ritland
>  -0.01602399
```

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

	b	b_log	Sp	Sp_log
obs_value	-0.0007007	-0.0357734	0.0007693	0.0392760
mean_sim	0.0000020	0.0000347	-0.0000008	0.0000097
sd_sim	0.0002752	0.0062994	0.0002726	0.0062438
0.95_inf	-0.0006246	-0.0141703	-0.0004583	-0.0098627
0.95_sup	0.0004646	0.0100574	0.0006179	0.0141312
0.9_inf	-0.0004780	-0.0112594	-0.0004014	-0.0087134
0.9_sup	0.0004031	0.0089617	0.0004759	0.0112549
pval_upper	0.0150000	0.0000000	0.9890000	1.0000000
pval_lower	0.9850000	1.0000000	0.0110000	0.0000000
pval_2sides	0.0300000	0.0000000	0.0220000	0.0000000

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

G.2 Clonal subrange

Basic commands:

```r
clonal_sub(posidonia, coords = coord_posidonia)
```

or, with MLL:

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

or, for haploid data:

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

Options: same distance classes definition as *autocorrelation*:

```r
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(posidonia, coords = coord_posidonia, vecdist = distvec)
```

```
                                                     #custom distance classes
clonal_sub(posidonia, coords = coord_posidonia, class1 = TRUE, d = 7)
                                                     #7 equidistant classes
clonal_sub(posidonia, coords = coord_posidonia, class1 = TRUE, d = 7)
                  #7 distance classes with the same number of units in each
```

Results:

```
res <- clonal_sub(posidonia, coords = coord_posidonia)
res[[1]] #Global clonal subrange
```

```
> [1] 11.6619
```

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

nb_pairs	dist_min	dist_max	dist_mean	Fr	log(Fr)
97	0.5	7.516648	4.683712	0.1649485	-0.7826518
157	7.615773	15.20691	11.14811	0.04458599	-1.350802
119	15.23975	22.80351	18.80791	0	-Inf
110	22.94014	30.41381	26.64826	0	-Inf
121	30.5	38.00329	34.2065	0	-Inf
64	38.02959	45.59879	41.52415	0	-Inf
34	46.09772	53.08484	49.56856	0	-Inf
29	53.53737	60.66144	57.05583	0	-Inf
31	61.00205	68.00184	64.65715	0	-Inf
18	68.52919	76.04111	70.91218	0	-Inf

G.3 Aggregation index

Basic commands:

```
agg_index(posidonia, coords = coord_posidonia)
```

or, with MLL:

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

or, for haploid data:

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

Options:

```
agg_index(posidonia, coords = coord_posidonia, nbrepeat = 100) #pvalue computation
agg_index(posidonia, coords = coord_posidonia, nbrepeat = 1000, bar = TRUE)
                                                            #could be time consuming
```

Results:

```
res <- agg_index(posidonia, coords = coord_posidonia, nbrepeat = 1000)
```

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

Ac	pval	nbrepeat
0.2272127	0	1000

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

G.4 Edge Effect

Basic commands:

```
#for posidonia, center of quadra is at 40,10
edge_effect(posidonia, coords = coord_posidonia, center = c(40,10))
```

or, with MLL:

```
edge_effect(popsim, coords = coord_sim, center = c(40,10), listMLL = MLLlist)
```

or, for haploid data:

```
edge_effect(haplodata, coords = coord_haplo, center = c(40,10))
```

Options:

```
edge_effect(posidonia, coords = coord_posidonia, center = c(40,10), nbrepeat = 100)
                                                           #pvalue computation
edge_effect(posidonia, coords = coord_posidonia, center = c(40,10), nbrepeat = 1000,
                                                bar = TRUE) #could be time consuming
```

Results:

```
res <- edge_effect(posidonia, coords = coord_posidonia, center = c(40,10), nbrepeat = 1000)
```

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

Ee	pval_Ee	nbrepeat
0.0778672	0.434	1000

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

H. BONUS

Summary function:

Basic commands:

```
genclone(posidonia, coords = coord_posidonia)
```

or, with MLL:

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

or, for haploid data:

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

Options:

```
genclone(posidonia, coords = coord_posidonia, nbrepeat = 100) #pvalues
genclone(posidonia, coords = coord_posidonia, nbrepeat = 1000, bar = TRUE)
                                               #could be time consuming
```

Results:

```
genclone(posidonia, coords = coord_posidonia)
```

N	Lineage	nb_L	nb_all	SE	Fis	pval_2sides	Fis_WR	pval_2sides.1	R
40	MLG	28	4.142857	0.7693093	0.05076926	NA	0.02568129	NA	0.6923077

Pareto_index	Sp_Loiselle	pval_2sides	Sp_L_WR	pval_2sides.1	Sp_Ritland	pval_2sides.2
1.180756	0.001230855	NA	0.0012436	NA	0.0007693264	NA

Sp_R_WR	pval_2sides	H"	J'	D	V	Hill
0.0008031684	NA	3.149621	0.9452064	0.9705128	0.7921811	33.91304