Package ‘RClone’

August 29, 2016

Version 1.0.2

Date 2016-05-31

Title Partially Clonal Populations Analysis

Author Sophie Arnaud-Haond [aut],
Diane Bailleul [aut, cre],
Solenn Stoeckel [ctb]

Maintainer Diane Bailleul <diane.bailleul.pro@gmail.com>

Description R version of 'GenClone' (a computer pro-
gram to analyse genotypic data, test for clonality and describe spatial clonal organization, Arnaud-
Haond & Belkhir 2007, <http://wwz.ifremer.fr/clonix/content/download/68205/903914/file/GenClone2.0.setup.zip>), this p
age allows clone handling as 'GenClone' does, plus the possibility to work with several popula-
tions, MultiLocus Lineages (MLL) custom definition and use, and p-value calcula-
tion for psex statistic (probability of originating from distinct sexual events) and psex_Fis statis-
tic (taking account of Hardy-Weinberg equilibrium departure) as 'MLGsim'/'MLGsim2' (a pro-
gram for detecting clones using a simulation approach, Stenberg et al. 2003).

License GPL (>= 2.0)

LazyLoad yes

Depends R (>= 3.2.0), graphics, stats, grDevices, utils, datasets,
methods

URL https://github.com/dbailleul/RClone

Suggests knitr, pander, testthat

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2016-06-06 12:57:15

R topics documented:

`RClone-package` *RClone*

Description

`RClone` is a R package gathering all the functions of GenClone program to handle data (haploid and diploid) with clones.

Details

Package:	RClone
Type:	Package
Title:	GenClone
Version:	1.0
Date:	2015-07-31
License:	GPL (>=2.0)
LazyLoad:	yes

This package contains several types of functions:

- import/export functions to handle data from GenClone (see `transcript_GC`) and Adegenet and export `RClone` data for Genetix and Arlequin (for example see `export_genclone_genetix`),
- functions to help defining MLL (MultiLocus Lineage) as `psex` and `genet_dist`,
- descriptive functions to compute genotypic richness and diversity: `clonal_index`, `genclone` and `Pareto_index`,
- functions for spatial analyses of clonal structure (see for example `autocorrelation`).

Author(s)

Creator/Author: Diane Bailleul <diane.bailleul.pro@gmail.com>
Author: Sophie Arnaud-Haond <sophie.arnaud@ifremer.fr>
Contributor: Solenn Stoeckel

The R implementation of `RClone` was written by Diane Bailleul.

The design was inspired by GenClone program described in Arnaud-Haond & Belkhir (2007).

References

Review: "Standardizing methods to address clonality in population studies" 2007, Molecular Ecology, S. Arnaud-Haond, C.M. Duarte, F. Alberto and E.A. Serrao

---

`agg_index` *Aggregation of clones*

Description

`agg_index` computes `Ac` (aggregation of clonal lineages) assessed by comparing the probability of clonal identity between nearest units pairs.

Usage

```
agg_index(data1, coords = NULL, vecpop = NULL, nbrepeat = 1, bar = FALSE,
listMLL = NULL)
```

Arguments

`data1`	a `Rclone` table with one allele per column, haploid or diploid data.
`coords`	a table with coordinates of every units in `data1`.
`vecpop`	vector, option, `vecpop` indicates the population name of each unit of `data1`, if `data1` contains several populations. If `data1` contains only one population, leave `vecpop = NULL`.
`nbrepeat`	numeric, the number of repeats.
`bar`	option, if `TRUE`, adds a progression bar.
`listMLL`	option, a custom list of MLL.

Details

The probability of clonal identity is set as 0 if ramets belong to the same MLG/MLL and 1 otherwise.

`Ac` is computed as `Ac=(Psg-Psp)/Psg` with `Psg` the average probability of clonal identity of all pairs and `Psp` among pairwise nearest neighbours.

Coordinates of units are randomly permuted `nbrepeat` times to provide a upper pvalue for `Ac` (Monte Carlo).

Value

a list (one population) or a list of lists (multi-populations) of:

- results a table with `Ac` value, pvalue and the number of permutations.
- simulations a vector of `nbrepeat` values of `sim-Ac`.

Author(s)

Creator/Author: Diane Bailleul <diane.bailleul.pro@gmail.com>
Author: Sophie Arnaud-Haond <sophie.arnaud@ifremer.fr>
Contributor: Solenn Stoeckel

The R implementation of `RClone` was written by Diane Bailleul.

The design was inspired by GenClone program described in Arnaud-Haond & Belkhir (2007).

References

Arnaud-Haond et al., 2007, Standardizing methods to address clonality in population studies.

See Also

`autocorrelation`, `clonal_sub` and `edge_effect`

Examples

```
data(posidonia)
data(coord_posidonia)

agg_index(posidonia, coords = coord_posidonia)
#agg_index(posidonia, coords = coord_posidonia, nbrepeat = 1000, bar = TRUE) #takes time
```

`autocorrelation` *Spatial Autocorrelation*

Description

`autocorrelation` computes kinship coefficients (Loiselle or Ritland) between pairs of individuals within specific ranges of geographic distance.

Usage

```
autocorrelation(data1, haploid = FALSE, coords = NULL, vecpop = NULL, listMLL = NULL,
Loiselle = FALSE, Ritland = FALSE,
genet = FALSE, central_coords = FALSE, random_unit = FALSE, weighted = FALSE,
class1 = FALSE, class2 = FALSE, d = NULL, vecdist = NULL,
graph = FALSE, nbrepeat = NULL, export = FALSE)
```

Arguments

`data1`	a `Rclone` table with one allele per column.
`haploid`	logical, option, `haploid` indicates the ploidy level of `data1`.
`coords`	a table with coordinates of every units in `data1`.
`vecpop`	vector, option, `vecpop` indicates the population name of each unit of `data1`, if `data1` contains several populations. If `data1` contains only one population, leave `vecpop = NULL`.
`listMLL`	option, a custom list of MLL.
`Loiselle`	logical, if `TRUE`, Loiselle kinship coefficients are computed.
`Ritland`	logical, if `TRUE`, Ritland kinship coefficients are computed.
`genet`	option, `TRUE` keeps only MLG of `data1`.
`central_coords`	option, if `genet = TRUE`, `central_coords` computes central coordinates for each MLG/MLL.
`random_unit`	option, if `genet = TRUE`, `random_unit` keeps coordinates of only one unit per MLG/MLL.
`weighted`	option, if `genet = TRUE`, `weighted` computes a weighted matrix over ramets.
`class1`	option, if `TRUE`, computes distance classes of `d` equidistant classes.
`class2`	option, if `TRUE`, computes distance classes of `d` classes with the same number of units pairs each.
`d`	numeric, option, number of distance classes. By default, d = 10.
`vecdist`	option, a custom vector distance to construct distance classes.
`graph`	option, if `TRUE`, displays kinship coefficient between pairs plotted against distance.
`nbrepeat`	numeric, option, if `pvalue = TRUE`, `nbrepeat` is the number of resampling to enable pvalues computation.
`export`	option, if `TRUE`, graph is saved as .eps into working directory.

Details

By default, `d = 10` and `autocorrelation` computes 10 equidistant distance classes for all the ramets pairs.
The function proposes 3 others options:

- `class1` fixing `d` equidistant classes,
- `class2` fixing `d` distance classes with the same number of units pairs,
- `maxdist = TRUE` allowing the user to give a vector `vecdist` of intervals.

The function computes one of the two average kinship coefficients: Loiselle and Ritland.

Autocorrelation can be compute on ramets level, or genet level with:

- central coordinates of each MLG/MLL,
- a re-sampling approach which randomly allocates one of the unit's coordinates per MLG/MLL (Alberto 2005),
- keeping all the ramets but weighting the matrix distances by a weighted matrix (Wagner 2005) where units of the same MLG/MLL are set to 0.

A permutation approach could be perform to assess pvalue and confidence intervals by permutation of the geographic coordinates among units.
For the re-sampling approach, a unit of each MLG/MLL is randomly picked at each permutation.
The p-value of mean kinship coefficients is related with the overall mean kinship coefficient: upper p-value (Monte Carlo) if greater or equal to the overall; otherwise, lower p-value.
For `b` and `Sp`, their p-value correspond to upper p-value.

Value

`autocorrelation` returns a list (one population) or lists of list (several populations) of:

- `Main_results`, a table with for each class, min, max, mean and ln(mean) of distance between two units, the number of pairs, the mean kinship coefficient and
  if `pvalue = TRUE`, the pvalue.
- `Slope_and_Sp_index`, a table with slopes of the regression between genetic and geographic/log(geographic) distances and `Sp` and `Sp_log` (used to quantify Spatial Genetic Structure, Vekemans and Hardy, 2004) as observed values, mean and
  standard deviation of the simulated values, 95% and 90% confidence intervals and p-value.
- `Slope_resample`, a table with slopes of the regression between genetic and geographic/log(geographic) distances at each pvalue.
- `Kinship_resample`, a table with for each class in column and each pvalue in row the mean kinship coefficient.
- `Matrix_kinship_results`, a dist object with kinship coefficients.
- `Class_kinship_results`, a list of kinship coefficients by distance class.
- `Class_distance_results`, a list of geographical distances by distance class.

Author(s)

Creator/Author: Diane Bailleul <diane.bailleul.pro@gmail.com>
Author: Sophie Arnaud-Haond <sophie.arnaud@ifremer.fr>
Contributor: Solenn Stoeckel

The R implementation of `RClone` was written by Diane Bailleul.

The design was inspired by GenClone program described in Arnaud-Haond & Belkhir (2007).

References

Loiselle et al., 1995, Spatial genetic structure of a tropical understory shrub, *Psychotria officinalis* (Rubiaceae).

Ritland, 1996, A marker-based method for inferences about quantitative inheritance in natural populations.

Arnaud-Haond et al., 2007, Standardizing methods to address clonality in population studies.

Vekemans & Hardy, 2004, New insights from fine-scale spatial genetic structure analyses in plant populations.

See Also

`kinship_Loiselle`, `kinship_Ritland`

Examples

```
data(posidonia)
data(coord_posidonia)

distGC <- c(0,10,15,20,30,50,70,76.0411073)

#res1 <- autocorrelation(posidonia, coords = coord_posidonia, Loiselle = TRUE, nbrepeat = 1000)

#res2 <- autocorrelation(posidonia, coords = coord_posidonia, Loiselle = TRUE,
#class2 = TRUE, d = 7)

#res2[[1]] #Main_results
#res1[[2]] #Slope_and_Sp_index
#res2[[3]] #Slope_and_Sp_index

#res3 <- autocorrelation(posidonia, coords = coord_posidonia, Loiselle = TRUE,
#vecdist = distGC, graph = TRUE)
```

`clonal_index` *Indices for clonal data*

Description

`clonal_index` computes main genotypic diversity and richness indices.

Usage

```
clonal_index(data1, vecpop = NULL, listMLL = NULL)
```

Arguments

`data1`	a `RClone` table with one allele per column, haploid or diploid data.
`vecpop`	vector, option, `vecpop` indicates the population name of each unit of `data1`, if `data1` contains several populations. If `data1` contains only one population, leave `vecpop = NULL`.
`listMLL`	option, a custom list of MLL.

Details

`clonal_index` returns:

- the number of units `N`,
- the number of unique genotypes `G`,
- the clonal diversity index `R` (Dorken & Eckert 2001; Ellstrand & Roose 1987),
- the Shannon-Wiener index estimator `H''` (Pielou 1966),
- the Pielou evenness index `J'` (Pielou 1975),
- the Simpson complement unbiased `D'` (Pielou 1969 ; Gini 1912 ; Peet 1974),
- the Simpson complement index `V` (Hurlbert 1971 ; Fager 1972),
- the reciprocal of Simpson index unbiased `Hill` (Hurlbert 1971 ; Hill 1973).

Value

a table (one population) or a list of tables (several population) with genotypic indices.

Author(s)

Creator/Author: Diane Bailleul <diane.bailleul.pro@gmail.com>
Author: Sophie Arnaud-Haond <sophie.arnaud@ifremer.fr>
Contributor: Solenn Stoeckel

The R implementation of `RClone` was written by Diane Bailleul.

The design was inspired by GenClone program described in Arnaud-Haond & Belkhir (2007).

References

Arnaud-Haond et al., 2007, Standardizing methods to address clonality in population studies.

See Also

`Pareto_index`

Examples

```
data(posidonia)

clonal_index(posidonia)
```

---

clonal_sub *Clonal Subrange*

Description

`clonal_sub` computes the clonal subrange analysis with spatial distance intervals and the corresponding probabilities of clonal identity.

Usage

```
clonal_sub(data1, coords = NULL, vecpop = NULL, listMLL = NULL, class1 = FALSE,
class2 = FALSE, d = NULL, vecdist = NULL)
```

Arguments

`data1`	a `Rclone` table with one allele per column, haploid or diploid data.
`coords`	a table with coordinates of every units in `data1`.
`vecpop`	vector, option, `vecpop` indicates the population name of each unit of `data1`, if `data1` contains several populations. If `data1` contains only one population, leave `vecpop = NULL`.
`listMLL`	option, a custom list of MLL.
`class1`	option, if `TRUE`, computes distance classes of `d` equidistant classes.
`class2`	option, if `TRUE`, computes distance classes of `d` classes with the same number of units pairs each.
`d`	numeric, number of distance classes.
`vecdist`	option, a custom vector distance intervals to construct distance classes.

Details

By default, `d = 10` and `clonal_sub` computes 10 equidistant distance classes for all the ramets pairs.

The function proposes 3 others options:

- `class1` fixing `d` equidistant classes,
- `class2` fixing `d` distance classes with the same number of units pairs,
- `vecdist != NULL` allowing the user to give a vector, `vecdist` of intervals. `vecdist` must start with 0 and end with `max(dist)`.

Value

A list of:

`clonal_sub_res`	clonal subrange, i.e. maximum distance between two units sharing the same MLG/MLL (Alberto et al., 2005)
`clonal_sub_tab`	table of results with, per class, the number of units pairs, the min, max and mean distances between pairs and `Fr/log(Fr)` the fraction of pairs of ramets sharing the same MLG/MLL

For multi-population `data1`, a list of lists per population.

Author(s)

Creator/Author: Diane Bailleul <diane.bailleul.pro@gmail.com>
Author: Sophie Arnaud-Haond <sophie.arnaud@ifremer.fr>
Contributor: Solenn Stoeckel

The R implementation of `RClone` was written by Diane Bailleul.

The design was inspired by GenClone program described in Arnaud-Haond & Belkhir (2007).

References

Alberto et al., 2005, Spatial genetic structure, neighbourhood size and clonal subrange in seagrass (*Cymodocea nodosa*) populations.

Arnaud-Haond et al., 2007, Standardizing methods to address clonality in population studies.

See Also

`autocorrelation`, `agg_index` and `edge_effect`

Examples

```
data(posidonia)
data(coord_posidonia)

distGC <- c(0,10,15,20,30,50,70,76.0411073)

clonal_sub(posidonia, coords = coord_posidonia)
clonal_sub(posidonia, coords = coord_posidonia, vecdist = distGC)
```

`convert_GC` *File conversion into RClone files*

Description

`convert_GC` helps files conversion into `RClone` format.

`RClone` functions work on tables with one allele per column.

`convert_GC` converts tables with one locus per column into tables with one allele per column, handling separation elements.

`convert_GC` also sorts alleles at a locus per increasing order.

Usage

```
convert_GC(data1, num, ele)
```

Arguments

`data1`	a table, with units in row and one locus per column.
`num`	numeric, the length of each allele.
`ele`	option, the alleles separator in the original table.

Value

a table with one allele per column, alleles sorted by increasing order.

Author(s)

Creator/Author: Diane Bailleul <diane.bailleul.pro@gmail.com>
Author: Sophie Arnaud-Haond <sophie.arnaud@ifremer.fr>
Contributor: Solenn Stoeckel

The R implementation of `RClone` was written by Diane Bailleul.

The design was inspired by GenClone program described in Arnaud-Haond & Belkhir (2007).

See Also

`sort_all` for tables with one allele per column.
`transcript_GC` uses `convert_GC` as internal function.

Examples

```
test <- matrix("232/231", ncol = 2, nrow = 2)
convert_GC(test, 3, "/")
#"232" is a allele of length 3 and "/" is the separator.

test2 <- matrix("192235", ncol = 2, nrow = 2)
convert_GC(test2, 3)
```

```
#no separator

#with data1, a genind object from adegenet:
#test <- genind2df(data1)
#convert_GC(test, 3, "/")
```

`coord_posidonia` *Posidonia coordinates*

Description

The quadra coordinates of a sub-dataset of *Posidonia oceanica* sampled in Mediterranean sea.

Usage

```
data("coord_posidonia")
```

Format

A data frame with 40 observations on the following 2 variables:

x a numeric vector, x-coordinate

y a numeric vector, y-coordinate

Source

Arnaud-Haond S, Alberto F, Eguiluz VM, Hernandez-Garcia E, Duarte CM, Serrao EA (2014) Disentangling the influence of mutation and migration in clonal seagrasses using the Genetic Distance Spectrum for microsatellites.

Dryad Digital Repository: http://dx.doi.org/10.5061/dryad.3b8k6

References

Arnaud-Haond S, Maolic Y, Hernandez-Garcia E, Eguiluz VM, Alberto F, Serrao EA, Duarte CM (2014) Disentangling the influence of mutation and migration in clonal seagrasses using the Genetic Distance Spectrum for microsatellites. Journal of Heredity 105(4): 532-541. http://dx.doi.org/10.1093/jhered/esu015

Arnaud-Haond S & Belkhir K, 2007, GENCLONE: a computer program to analyse genotypic data, test for clonality and describe spatial clonal organization.

Examples

```
data(coord_posidonia)
```

edge_effect *Edge Effect*

Description

edge_effect tests the occurrence of Edge Effect.

Usage

```
edge_effect(data1, coords = NULL, center = NULL, vecpop = NULL, nbrepeat = 1,
bar = FALSE, listMLL = NULL)
```

Arguments

data1	a Rclone table with one allele per column, haploid or diploid data.
coords	a table with coordinates of every units in data1.
center	a vector or a list of vectors, with c(x,y) coordinates of the centre of the sampling area.
vecpop	vector, option, vecpop indicates the population name of each unit of data1, if data1 contains several populations. If data1 contains only one population, leave vecpop = NULL.
nbrepeat	numeric, option, the number of repeats.
bar	logical, option, if TRUE, adds a progression bar.
listMLL	option, a custom list of MLL.

Details

The index of edge effect Ee estimates the effect of sampling (scheme and strategy) on genotypic richness estimation and in particular overestimation due to large clones sampled only once at the edge of the sampling area.

Ee is estimated as Ee=(Du-Da)/Da with Du average geographic distances between unique MLG/MLL and the centre, and Da between all sampling units and the centre.

As for the aggregation index Ac, coordinates of units are randomly permuted nbrepeat times to provide a upper p-value (Monte Carlo).

Value

a list (one population) or list of lists (several populations) with

- results a table with Ee value, pvalue and the number of permutations.
- simulations a vector of nbrepeat values of sim-Ee.

Author(s)

Creator/Author: Diane Bailleul <diane.bailleul.pro@gmail.com>
Author: Sophie Arnaud-Haond <sophie.arnaud@ifremer.fr>
Contributor: Solenn Stoeckel

The R implementation of `RClone` was written by Diane Bailleul.

The design was inspired by GenClone program described in Arnaud-Haond & Belkhir (2007).

References

Arnaud-Haond et al., 2007, Standardizing methods to address clonality in population studies.

See Also

`autocorrelation`, `clonal_sub` and `agg_index`

Examples

```
data(posidonia)
data(coord_posidonia)

center1 <- c(40,10)
#Our sample quadra ranges from 0 to 80 and 0 to 20

edge_effect(posidonia, coords = coord_posidonia, center = center1, nbrepeat = 1000,
bar = TRUE)

#But if, for some reasons you don't know where the middle of the sampling
##area is, you can try some of these:
center <- c(mean(coord_posidonia[,1]), mean(coord_posidonia[,2])) #or
center <- c(mean(c(min(coord_posidonia[,1]), max(coord_posidonia[,1]))),
mean(c(min(coord_posidonia[,2]), max(coord_posidonia[,2])))) #or
center <- c((max(coord_posidonia[,1])-min(coord_posidonia[,1]))/2,
(max(coord_posidonia[,2])-min(coord_posidonia[,2]))/2)
```

`export_genclone` *export data file to Adegenet, Genetix and Arlequin*

Description

These functions allow to transform a `RClone` table into files to work with `Adegenet` (R package), Genetix and Arlequin softwares.

Usage

```
export_genclone_genind(data1, ele)
export_genclone_genetix(data1, haploid = FALSE, ele, name)
export_genclone_arlequin(data1, haploid = FALSE, name)
```

Arguments

`data1`	a `RClone` table with only one population.
`haploid`	logical, option, if `haploid = FALSE`, `data1` contains diploid data; if `haploid` = TRUE, haploid data.
`ele`	option, separator element for export.
`name`	option, name of the exported file.

Value

a `genind` object or a file for Genetix or Arlequin.

Note

For multi-population files, we recommend to use split function to cut the table into several tables, one for each population, and then combine `lapply` with the export functions.

Author(s)

Creator/Author: Diane Bailleul <diane.bailleul.pro@gmail.com>
Author: Sophie Arnaud-Haond <sophie.arnaud@ifremer.fr>
Contributor: Solenn Stoeckel

The R implementation of `RClone` was written by Diane Bailleul.

The design was inspired by GenClone program described in Arnaud-Haond & Belkhir (2007).

Examples

```
data(posidonia)

#RClone to Adegenet:
res <- export_genclone_genind(posidonia, "/")
#library(adegenet)
#res2 <- df2genind(res, ploidy = 2, sep = "/")
#nAll(res2)

#RClone to Genetix:
export_genclone_genetix(posidonia, name = "test.txt")
#or
write.table(export_genclone_genetix(posidonia), "test2.txt", row.names = FALSE,
sep = "\t", quote = FALSE)
#for genets only:
export_genclone_genetix(unique(posidonia), name = "test.txt")

#Rclone to Arlequin:
write.table(export_genclone_arlequin(posidonia), "file1.arp", row.names = FALSE,
 col.names = FALSE, quote = FALSE)
#or
export_genclone_arlequin(posidonia, haploid = FALSE, "file2.arp")
#for genets only:
```

```
export_genclone_arlequin(unique(posidonia), haploid = FALSE, "file2.arp")

#if several populations:
#res <- split(data, vecpop)
#lapply(res, function(x) export_genclone_genetix(x))
#lapply(res, function(x) export_genclone_arlequin(x))
```

`Fis` *Fis*

Description

`Fis` computes observed Heterozygosity (`Hobs`), expected Heterozygosity (`Hexp`; Nei, 1978) and `Fis` from ramets or genets.

Usage

```
Fis(data1, vecpop, genet = FALSE, RR = FALSE)
```

Arguments

`data1`	a `Rclone` table with one allele per column for diploid data.
`vecpop`	vector, option, `vecpop` indicates the population name of each unit of `data1`, if `data1` contains several populations. If `data1` contains only one population, leave `vecpop = NULL`.
`genet`	option, if `TRUE`, `data1` is reduced to genets.
`RR`	option, if `TRUE`, `Fis` and allelic frequencies are computed with Round-Robin method.

Details

Allelic frequencies are computed:

- on ramet level,
- on genet level (`genet = TRUE`),
- with Round-Robin method (`RR = TRUE`, see `freq_RR`).

Value

a table with `Hobs`, `Hexp` and `Fis` for each locus.

If `RR = TRUE`, a list of the `Hobs/Hexp/Fis` table and another table with Round-Robin frequencies.

If `data1` is a multi-population table, a list of table(s) for each population.

Author(s)

Creator/Author: Diane Bailleul <diane.bailleul.pro@gmail.com>
Author: Sophie Arnaud-Haond <sophie.arnaud@ifremer.fr>
Contributor: Solenn Stoeckel

The R implementation of `RClone` was written by Diane Bailleul.

The design was inspired by GenClone program described in Arnaud-Haond & Belkhir (2007).

References

Arnaud-Haond et al., 2007, Standardizing methods to address clonality in population studies.

See Also

`freq_RR`, `pgen`, `pgen_Fis`, `psex` and `psex_Fis`

Examples

```
data(posidonia)

Fis(posidonia)
Fis(posidonia, genet = TRUE)
Fis(posidonia, RR = TRUE)
```

`freq_RR` *Allelic Frequencies*

Description

`freq_RR` returns a table of allelic frequencies computed with or without Round-Robin method.

Usage

```
freq_RR(data1, haploid = FALSE, vecpop = NULL, genet = FALSE, RR = FALSE)
```

Arguments

`data1`	a `Rclone` table with one allele per column.
`haploid`	logical, option,`haploid` indicates the ploidy level of `data1`.
`vecpop`	vector, option, `vecpop` indicates the population name of each unit of `data1`, if `data1` contains several populations. If `data1` contains only one population, leave `vecpop = NULL`.
`genet`	option, if `TRUE`, `data1` is reduced to genets.
`RR`	option, if `TRUE`, indicates frequencies are computed with Round-Robin method.

Details

Round-Robin method (Parks & Werth 1993) is a sub-sampling approach which avoids overestimation of rare alleles.
Each locus frequency is estimated on MLG lists constructed without the locus sampled.
This calculation is repeated for all loci.

Value

a table (one population) or a list of tables (several populations) with three columns:

- a first column with the number of the locus considered (written as *"locus_1")*,
- a second column with the list of the unique alleles of the locus,
- a last column with the frequency of the allele in row.

Author(s)

Creator/Author: Diane Bailleul <diane.bailleul.pro@gmail.com>
Author: Sophie Arnaud-Haond <sophie.arnaud@ifremer.fr>
Contributor: Solenn Stoeckel

The R implementation of `RClone` was written by Diane Bailleul.

The design was inspired by GenClone program described in Arnaud-Haond & Belkhir (2007).

References

Parks & Werth, 1993, A study of spatial features of clones in a population of Bracken fern, *Pteridium aquilinum* (Dennstaedtiaceae). Arnaud-Haond et al., 2007, Standardizing methods to address clonality in population studies.

See Also

`pgen` and `pgen_Fis`

Examples

```
data(posidonia)

freq_RR(posidonia, RR = TRUE)
freq_RR(posidonia)
```

`GenClone`	*Summary function of RClone package*

Description

`genclone` computes main genetic/genotypic diversity/richness indices.

Usage

```
GenClone(data1, haploid = FALSE, coords = NULL, vecpop = NULL, listMLL = NULL,
nbrepeat = NULL, bar = FALSE)
```

Arguments

`data1`	a `Rclone` table with one allele per column.
`haploid`	logical, option, `haploid` indicates the ploidy level of `data1`.
`coords`	a table with coordinates of every units in `data1`.
`vecpop`	vector, option, `vecpop` indicates the population name of each unit of `data1`, if `data1` contains several populations. If `data1` contains only one population, leave `vecpop = NULL`.
`nbrepeat`	numeric, option, if `pvalue = TRUE`, `nbrepeat` is the number of resampling to enable pvalues computation.
`listMLL`	option, a custom list of MLL.
`bar`	option, if `TRUE`, displays a progression bar.

Details

`GenClone` returns results of several functions of `RClone`: a summary of `MLG_tab`, `Fis` on ramets and genets with pvalues (resample the population `nbrepeat` times, with simulated sexual events), `B_Pareto` from `Pareto_index`, `Sp` from `autocorrelation` and indexes from `clonal_index`.

If no coordinate at all are available, let `coords = NULL` as it or create a table with always the same number (i.e. "999", "-1", etc.). If coordinates are available for some populations only, for the population with missing coordinates: replace all the coordinates by the same number, as "999". `GenClone` cannot handle mix situation with missing coordinates only for some units of the population.

Value

`GenClone` returns a table with:

- N, the number of units in data1,
- Lineage, MLG or MLL,
- nb_L, the number of MLG/MLL,
- nb_all, the mean number of alleles,

- SE, the standard error of `nb_all`,
- Fis, on ramets if diploid data
- pval_2sides, the two-sided p-value of `Fis` if `nbrepeat`,
- Fis_WR, on genets if diploid data
- pval_2sides, the two-sided p-value of `Fis_WR` if `nbrepeat`,
- R, the clonal diversity index (Dorken & Eckert 2001; Ellstrand & Roose 1987),
- Pareto_index, the index of Pareto
- Sp_Loiselle, `Sp` index computed on ramets with Loiselle kinship results used to quantify Spatial Genetic Structure (Vekemans and Hardy, 2004)
- pval_2sides, the two-sided p-value of `Sp_Loiselle` if `nbrepeat`,
- Sp_Ritland, `Sp` index computed on ramets with Ritland kinship results used to quantify SGS
- pval_2sides, the two-sided p-value of `Sp_Ritland` if `nbrepeat`,
- Sp_L_WR, `Sp` index computed on genets with Loiselles kinship results used to quantify SGS
- pval_2sides, the two-sided p-value of `Sp_L_WR` if `nbrepeat`,
- Sp_R_WR, , `Sp` index computed on genets with Ritland kinship results used to quantify SGS
- pval_2sides, the two-sided p-value of `Sp_R_WR` if `nbrepeat`,
- H", the Shannon-Wiener index estimator (Pielou 1966),
- J', the Pielou evenness index(Pielou 1975),
- D', the Simpson complement unbiased (Pielou 1969 ; Gini 1912 ; Peet 1974),
- V, the Simpson complement index (Hurlbert 1971 ; Fager 1972),
- Hill, the reciprocal of Simpson index unbiased (Hurlbert 1971 ; Hill 1973).

Author(s)

Creator/Author: Diane Bailleul <diane.bailleul.pro@gmail.com>
Author: Sophie Arnaud-Haond <sophie.arnaud@ifremer.fr>
Contributor: Solenn Stoeckel

The R implementation of `RClone` was written by Diane Bailleul.

The design was inspired by GenClone program described in Arnaud-Haond & Belkhir (2007).

References

Arnaud-Haond et al., 2007, Standardizing methods to address clonality in population studies.

See Also

`clonal_index`

Examples

```
data(posidonia)
data(coord_posidonia)

#GenClone(posidonia) #without coordinates
#GenClone(posidonia, coords = coord_posidonia) #with coordinates
#GenClone(posidonia, coords = coord_posidonia, nbrepeat = 1000)
##time consuming
```

---

genet_dist *Genetic distance*

Description

Defining MLL (MultiLocus Lineage): ascertaining that each distinct MLG (MultiLocus Genotype) belongs to a distinct genet (Halkett et al., 2005a).

Usage

```
genet_dist(data1, haploid = FALSE, vecpop = NULL, manh = FALSE, manh_w = FALSE,
graph = FALSE, breaking = NULL, alpha1 = NULL, alpha2 = NULL, export = FALSE)
genet_dist_sim(data1, haploid = FALSE, vecpop = NULL, nbrepeat = 1000,
genet = FALSE, manh = FALSE, manh_w = FALSE, graph = FALSE, breaking = NULL,
export = FALSE)
```

Arguments

`data1`	a `Rclone` table with one allele per column.
`haploid`	logical, option, `haploid` indicates the ploidy level of `data1`.
`vecpop`	vector, option, `vecpop` indicates the population name of each unit of `data1`, if `data1` contains several populations. If `data1` contains only one population, leave `vecpop = NULL`.
`manh`	option, if `TRUE`, computes genetic distances among MLG in terms of divergence of microsatellites motifs (Rozenfeld et al., 2007).
`manh_w`	option, if `TRUE`, computes genetic distances among MLG in terms of weighted divergence of microsatellites motifs (Rozenfeld et al., 2007).
`graph`	option, if `TRUE`, displays a barplot with `breaking` and `pas` arguments.
`breaking`	numeric, option, if `breaking != NULL`, adds `breaks` argument for barplot as `breaks = seq(0, max, X)`, with `X`, the numerical value of `breaking`.
`alpha1`	numeric, option, if `alpha1` is not NULL, a vertical significativity line is added on graph at `alpha1`
`alpha2`	numeric, option, if `alpha2` is not NULL, a vertical significativity line is added on graph at `alpha2`.
`nbrepeat`	numeric, the number of repeats for simulation (i.e. reproduction event).
`genet`	option, if `FALSE`, selfing is taking into account in simulation through ramets.
`export`	option, if `TRUE`, graph is saved as .eps into working directory.

Details

`genet_dist` and `genet_dist_sim` help determining MLL, i.e. if slightly different MLG belong or not to the same lineage.

`genet_dist` computes genetic distances between pairs of units in terms of number of alleles (Chakraborty and Jin, 1993) by default.

If `manh = TRUE` or `manh_w = TRUE`, divergence of SSR motifs (Rozenfeld et al., 2007) is used as genetic distance.

These distance distributions help defining MLL with significativity of `alpha`: every pair under alpha could be ramets of a genet.

`genet_dist_sim` computes genetic distances but after a reproduction event between the units.

The simulated distance distribution allows to distinguish slightly differences due to somatic mutation or scoring errors by stacking the two distributions.

Value

`genet_dist` returns:

- `distance_matrix`, a `dist` object with genetic distances by pair of units.
- `potential_clones`, a table containing names and genetic distances of pairs of units under `alpha1` distribution or of maximal genetic distance of `alpha2`.
- `all_pairs`, a table containing names and genetic distances of every pairs of units.
- `sign`, the numeric value of `alpha1` or `alpha2`.

If `vecpop != NULL`, a list for every population.

`genet_dist_sim` returns a `dist` object of genetic distances by pair of units after a sexual reproduction event.

Author(s)

Creator/Author: Diane Bailleul <diane.bailleul.pro@gmail.com>
Author: Sophie Arnaud-Haond <sophie.arnaud@ifremer.fr>
Contributor: Solenn Stoeckel

The R implementation of `RClone` was written by Diane Bailleul.

The design was inspired by GenClone program described in Arnaud-Haond & Belkhir (2007).

References

Chakraborty & Jin, 1993, Determination of relatedness between individuals using DNA-fingerprinting.

Arnaud-Haond et al., 2007, Standardizing methods to address clonality in population studies.

Rozenfeld et al., 2007, Spectrum of genetic diversity and networks of clonal populations.

Examples

```
data(posidonia)

res <- genet_dist(posidonia, manh = TRUE, graph = TRUE, alpha1 = 0.05)

#Combining functions:
res1 <- genet_dist(posidonia, manh = TRUE)$distance_matrix
res2 <- genet_dist_sim_core(posidonia, nbrepeat = 100, manh = TRUE, genet = TRUE)$distance_matrix

p1 <- hist(res1, freq = FALSE, col = rgb(0,0.4,1,1), breaks = seq(0, max(res1), 2))
p2 <- hist(res2, freq = FALSE, col = rgb(0.7,0.9,1,0.5), breaks = seq(0, max(res2), 2))

limx <- max(max(res1), max(res2))
plot(p1, col = rgb(0,0.4,1,1), freq = FALSE, xlim = c(0,limx))
plot(p2, col = rgb(0.7,0.9,1,0.5), freq = FALSE, add = TRUE)

#Other way:
p1 <- as.data.frame(table(res1))
p2 <- as.data.frame(table(res2))
barplot(p1$Freq/sum(p1$Freq), col=rgb(0,0.4,1,1), axis.lty = 1,
names.arg = as.numeric(as.character(p1[,1])))
barplot(p2$Freq/sum(p2$Freq), col=rgb(0.7,0.9,1,0.5), add = TRUE)
title("Genetic distances between pairs of MLG")

#Adding a legend:
leg.txt <- c("original data","simulated data")
col <- c(rgb(0,0.4,1,1), rgb(0.7,0.9,1,0.5))
legend("topright", fill = col, leg.txt, plot = TRUE, bty = "o", box.lwd = 1.5,
bg = "white")
```

`infile` *Infile GenClone style file*

Description

A GenClone file of 40 units of *Posidonia oceanica* (genotypes of seven loci and x/y coordinates) sampled in Mediterranean sea.

Usage

```
data("infile")
```

Format

A data frame with 41 observations on the following 12 variables (not relevant).

`V1` a numeric vector

`V2` a numeric vector

`V3` a numeric vector

V4 a numeric vector

V5 a numeric vector

V6 a factor with levels 208208 208210 208216 210212 210216 210218 212216 216218 222226 Po15

V7 a factor with levels 234234 234236 234242 Po5

V8 a factor with levels 159159 159163 159165 163163 163165 165165 Po5-49

V9 a factor with levels 168168 168170 168172 170170 170172 172172 Po5-40

V10 a factor with levels 178178 178180 180180 Po5-10

V11 a factor with levels Po4-3

V12 a factor with levels Po5-39

Details

This data is given as illustration of GenClone file formatted to work with RClone (the R package version of GenClone).

Source

Arnaud-Haond S, Alberto F, Eguiluz VM, Hernandez-Garcia E, Duarte CM, Serrao EA (2014) Disentangling the influence of mutation and migration in clonal seagrasses using the Genetic Distance Spectrum for microsatellites.

Dryad Digital Repository: http://dx.doi.org/10.5061/dryad.3b8k6

References

Arnaud-Haond S, Maolic Y, Hernandez-Garcia E, Eguiluz VM, Alberto F, Serrao EA, Duarte CM (2014) Disentangling the influence of mutation and migration in clonal seagrasses using the Genetic Distance Spectrum for microsatellites. Journal of Heredity 105(4): 532-541. http://dx.doi.org/10.1093/jhered/esu015

Arnaud-Haond S & Belkhir K, 2007, GENCLONE: a computer program to analyse genotypic data, test for clonality and describe spatial clonal organization.

Examples

```
data(infile)
#This is nearly a GenClone file, type:
#write.table(infile, "infile2.csv", col.names = FALSE, row.names = FALSE, sep = ";")
#Now you have a formatted GenClone file.
```

kinship — *Loiselle and Ritland kinship coefficients*

Description

`kinship_Loiselle` and `kinship_Ritland` compute average genetic distances or kinship coefficients.

Usage

```
kinship_Loiselle(data1, haploid = FALSE, vecpop = NULL)
kinship_Ritland(data1, haploid = FALSE, vecpop = NULL)
```

Arguments

`data1`	a `Rclone` table with one allele per column.
`haploid`	logical, option, `haploid` indicates the ploidy level of `data1`.
`vecpop`	vector, option, `vecpop` indicates the population name of each unit of `data1`, if `data1` contains several populations. If `data1` contains only one population, leave `vecpop = NULL`.

Value

a `dist` object (or a list of `dist` objects for multi-population `data1`) with genetic distances between pairs of units.

Author(s)

Creator/Author: Diane Bailleul <diane.bailleul.pro@gmail.com>
Author: Sophie Arnaud-Haond <sophie.arnaud@ifremer.fr>
Contributor: Solenn Stoeckel

The R implementation of `RClone` was written by Diane Bailleul.

The design was inspired by GenClone program described in Arnaud-Haond & Belkhir (2007).

References

Loiselle et al., 1995, Spatial genetic structure of a tropical understory shrub, *Psychotria officinalis* (Rubiaceae).

Ritland, 1996, A marker-based method for inferences about quantitative inheritance in natural populations.

Arnaud-Haond et al., 2007, Standardizing methods to address clonality in population studies.

See Also

`autocorrelation`

Examples

```
data(posidonia)

#kinship_Loiselle(posidonia)
#kinship_Ritland(posidonia)
```

---

list_all *Listing unique alleles*

Description

`list_all_tab` returns a table with loci in column and unique alleles in row.

Usage

```
list_all_tab(data1, haploid = FALSE, vecpop = NULL)
```

Arguments

`data1`	a `Rclone` table, with one allele per column.
`haploid`	logical, option, `haploid` indicates the ploidy level of `data1`.
`vecpop`	vector, option, `vecpop` indicates the population name of each unit of `data1`, if `data1` contains several populations. If `data1` contains only one population, leave `vecpop = NULL`.

Value

a table (one population) or a list of tables (several populations) with the unique alleles per locus.

Author(s)

Creator/Author: Diane Bailleul <diane.bailleul.pro@gmail.com>
Author: Sophie Arnaud-Haond <sophie.arnaud@ifremer.fr>
Contributor: Solenn Stoeckel

The R implementation of `RClone` was written by Diane Bailleul.

The design was inspired by GenClone program described in Arnaud-Haond & Belkhir (2007).

Examples

```
data(posidonia)

list_all_obj(posidonia, haploid = FALSE)
list_all_tab(posidonia, haploid = FALSE)
corresp_loci(posidonia, haploid = FALSE)
```

`MLG_tab` *Table of MLG (MultiLocus Genotypes)*

Description

`MLG_tab` returns a table with one row per MLG and several columns if there's several units per MLG.

Usage

```
MLG_tab(data1, vecpop = NULL)
```

Arguments

`data1`	a `Rclone` table with one allele per column, haploid or diploid data.
`vecpop`	vector, option, `vecpop` indicates the population name of each unit of `data1`, if `data1` contains several populations. If `data1` contains only one population, leave `vecpop = NULL`.

Value

a table (one population) or a list of tables (several populations)with one row per MLG and several columns if several units share the same MLG.

Author(s)

Creator/Author: Diane Bailleul <diane.bailleul.pro@gmail.com>
Author: Sophie Arnaud-Haond <sophie.arnaud@ifremer.fr>
Contributor: Solenn Stoeckel

The R implementation of `RClone` was written by Diane Bailleul.

The design was inspired by GenClone program described in Arnaud-Haond & Belkhir (2007).

References

Arnaud-Haond et al., 2007, Standardizing methods to address clonality in population studies.

Examples

```
data(posidonia)

res <- MLG_tab(posidonia)
res
```

MLL_generator *Clonal Lineage Generation*

Description

Defining MLL (MultiLocus Lineage): ascertaining that each distinct MLG (MultiLocus Genotype) belongs to a distinct genet (Halkett et al., 2005a).

Usage

```
MLL_generator(data1, haploid = FALSE, vecpop = NULL, manh = FALSE, manh_w = FALSE,
alpha1 = NULL, alpha2 = NULL)
MLL_generator2(potential_clones = NULL, res_mlg = NULL, vecpop = NULL)
```

Arguments

`data1`	a `Rclone` table with one allele per column.
`haploid`	logical, option, `haploid` indicates the ploidy level of `data1`.
`vecpop`	vector, option, `vecpop` indicates the population name of each unit of `data1`, if `data1` contains several populations. If `data1` contains only one population, leave `vecpop = NULL`.
`manh`	option, if `TRUE`, computes genetic distances among MLG in terms of divergence of microsatellites motifs (Rozenfeld et al., 2007).
`manh_w`	option, if `TRUE`, computes genetic distances among MLG in terms of weighted divergence of microsatellites motifs (Rozenfeld et al., 2007).
`alpha1`	numeric, option, if `alpha1` is not NULL, a vertical significativity line is added on graph at `alpha1`
`alpha2`	numeric, option, if `alpha2` is not NULL, a vertical significativity line is added on graph at `alpha2`.
`potential_clones`	table, a result table from `genet_dist` named `potential_clones`.
`res_mlg`	list, a list of MLG, result from `MLG_list`.

Details

`MLL_generator` creates automatically MLL from a given genetic distance (`alpha2`) or a percentage of the distribution of genetic distance (`alpha1`).

If several populations (`vecpop != NULL`), `MLL_generator` is the only function in the package `RClone` to accept different arguments for an option. `alpha1` and `alpha2` thus are vectors of several numeric values, one per populations.

If `manh = TRUE` or `manh_w = TRUE`, divergence of SSR motifs (Rozenfeld et al., 2007) is used as genetic distance.

`MLL_generator2` computes a list of MLL from previous results of `genet_dist` and `MLG_list`.

`MLL_generator` and `MLL_generator2` compute a list of MLL to use with others `RClone` functions.

Value

`MLL_generator` and `MLL_generator2` return a list of MLL (one population) or a list of lists (several populations).

Author(s)

Creator/Author: Diane Bailleul <diane.bailleul.pro@gmail.com>
Author: Sophie Arnaud-Haond <sophie.arnaud@ifremer.fr>
Contributor: Solenn Stoeckel

The R implementation of `RClone` was written by Diane Bailleul.

The design was inspired by GenClone program described in Arnaud-Haond & Belkhir (2007).

References

Chakraborty & Jin, 1993, Determination of relatedness between individuals using DNA-fingerprinting.

Arnaud-Haond et al., 2007, Standardizing methods to address clonality in population studies.

Rozenfeld et al., 2007, Spectrum of genetic diversity and networks of clonal populations.

See Also

`genet_dist`

Examples

```
data(popsim)

#MLLlist <- MLL_generator(popsim, alpha2 = 4)
#or
#res <- genet_dist(popsim, alpha2 = 4)
#MLLlist <- MLL_generator2(res$potential_clones, MLG_list(popsim))
#take few seconds
```

`Pareto_index` *Pareto index*

Description

`Pareto_index` computes parameters of the Pareto distribution.

Usage

```
Pareto_index(data1, vecpop = NULL, listMLL = NULL, full = FALSE, graph = FALSE,
legends = 1, export = FALSE)
```

Arguments

`data1`	a `Rclone` table with one allele per column, haploid or diploid data.
`vecpop`	vector, option, `vecpop` indicates the population name of each unit of `data1`, if `data1` contains several populations. If `data1` contains only one population, leave `vecpop = NULL`.
`listMLL`	option, a custom list of MLL.
`full`	option, if `TRUE`, gives more detailed results.
`graph`	option, if `TRUE`, displays plot of the inverse cumulated frequency of the number of lineages.
`export`	option, if `TRUE`, graph is saved as .eps into working directory.
`legends`	option, numerical, with `graph = TRUE`, `legends = 1` gives the log-log regression equation; `legends = 2` gives the Pareto index, the r2 and the p-value of the regression.

Details

Pareto's Beta is given as `-slope` of the linear regression of the inverse cumulated frequency of the number of lineages (Pareto 1897 in Vidondo 1997).

The distribution of clonal size in the population `c_Pareto` is computed as `slope+1` (Schroeder 1991).

Value

A list of:

`Pareto`	Pareto's Beta,
`c_Pareto`	distribution of clonal size in the population,
`coefficients and regression_results`	summary of the linear regression,
`coords_Pareto`	x and y coordinates of the inverse cumulated frequencies.

For several populations, a list of lists per population.

Author(s)

Creator/Author: Diane Bailleul <diane.bailleul.pro@gmail.com>
Author: Sophie Arnaud-Haond <sophie.arnaud@ifremer.fr>
Contributor: Solenn Stoeckel

The R implementation of `RClone` was written by Diane Bailleul.

The design was inspired by GenClone program described in Arnaud-Haond & Belkhir (2007).

References

Arnaud-Haond et al., 2007, Standardizing methods to address clonality in population studies.

See Also

`clonal_index`

Examples

```
data(posidonia)

Pareto_index(posidonia, graph = TRUE, legends = 2)

res <- Pareto_index(posidonia, full = TRUE)[[4]]

xi <- res[,1]
yi <- res[,2]
exp(summary(lm(log10(yi)~log10(xi)))$coefficients[1]) ##true b of y=ax+b
```

`pgen` *Probability of a Genotype*

Description

pgen and `pgen_Fis` compute the probability of a genotype under the Hardy-Weinberg equilibrium assumption (with or without taking account of departures from H-W equilibrium).

Usage

```
pgen(data1, haploid = FALSE, vecpop = NULL, genet = FALSE, RR = FALSE)
pgen_Fis(data1, vecpop = NULL, genet = FALSE, RR = FALSE)
```

Arguments

`data1`	a `Rclone` table with one allele per column.
`haploid`	logical, option, `haploid` indicates the ploidy level of `data1`. Not edible for `pgen_Fis`.
`vecpop`	vector, option, `vecpop` indicates the population name of each unit of `data1`, if `data1` contains several populations. If `data1` contains only one population, leave `vecpop = NULL`.
`genet`	option, if `genet = TRUE`, computes pgen on genet level.
`RR`	option, if `RR = TRUE`, computes pgen with Round-Robin method.

Value

a table (one population) or a list of tables (several populations) with pgen computed for each genotype.

Note

We strongly recommand to use `RR = TRUE` option to compute allelic frequencies for clonal data. Otherwise, we let the options to work with frequencies at genet level (`genet = TRUE`) or ramet level (`RR = FALSE` and `genet = FALSE`).

Author(s)

Creator/Author: Diane Bailleul <diane.bailleul.pro@gmail.com>
Author: Sophie Arnaud-Haond <sophie.arnaud@ifremer.fr>
Contributor: Solenn Stoeckel

The R implementation of `RClone` was written by Diane Bailleul.

The design was inspired by GenClone program described in Arnaud-Haond & Belkhir (2007).

References

Arnaud-Haond et al., 2007, Standardizing methods to address clonality in population studies.

See Also

`freq_RR`, `psex` and `psex_Fis`

Examples

```
data(posidonia)

pgen(posidonia, RR = TRUE)
pgen_Fis(posidonia, RR = TRUE)
```

`popsim` *Posidonia*

Description

A theorical diploid population of 100 units, 100 loci and 10 alleles max per locus, with c = 0.9999 (c, clonality rate) after 10000 generations.

Usage

```
data("popsim")
```

Format

A data frame with 100 observations on the following 200 variables.

`loc_1_1`	first allele of locus
`loc_1_2`	second allele of locus
`loc_2_1`	first allele of locus
`loc_2_2`	second allele of locus
`loc_3_1`	first allele of locus
`loc_3_2`	second allele of locus
`loc_4_1`	first allele of locus
`loc_4_2`	second allele of locus
`loc_5_1`	first allele of locus
`loc_5_2`	second allele of locus
`loc_6_1`	first allele of locus
`loc_6_2`	second allele of locus
`loc_7_1`	first allele of locus
`loc_7_2`	second allele of locus
`loc_8_1`	first allele of locus
`loc_8_2`	second allele of locus
`loc_9_1`	first allele of locus
`loc_9_2`	second allele of locus
`loc_10_1`	first allele of locus
`loc_10_2`	second allele of locus
`loc_11_1`	first allele of locus
`loc_11_2`	second allele of locus
`loc_12_1`	first allele of locus
`loc_12_2`	second allele of locus
`loc_13_1`	first allele of locus
`loc_13_2`	second allele of locus
`loc_14_1`	first allele of locus
`loc_14_2`	second allele of locus
`loc_15_1`	first allele of locus
`loc_15_2`	second allele of locus
`loc_16_1`	first allele of locus
`loc_16_2`	second allele of locus
`loc_17_1`	first allele of locus
`loc_17_2`	second allele of locus
`loc_18_1`	first allele of locus

`loc_18_2`	second allele of locus
`loc_19_1`	first allele of locus
`loc_19_2`	second allele of locus
`loc_20_1`	first allele of locus
`loc_20_2`	second allele of locus
`loc_21_1`	first allele of locus
`loc_21_2`	second allele of locus
`loc_22_1`	first allele of locus
`loc_22_2`	second allele of locus
`loc_23_1`	first allele of locus
`loc_23_2`	second allele of locus
`loc_24_1`	first allele of locus
`loc_24_2`	second allele of locus
`loc_25_1`	first allele of locus
`loc_25_2`	second allele of locus
`loc_26_1`	first allele of locus
`loc_26_2`	second allele of locus
`loc_27_1`	first allele of locus
`loc_27_2`	second allele of locus
`loc_28_1`	first allele of locus
`loc_28_2`	second allele of locus
`loc_29_1`	first allele of locus
`loc_29_2`	second allele of locus
`loc_30_1`	first allele of locus
`loc_30_2`	second allele of locus
`loc_31_1`	first allele of locus
`loc_31_2`	second allele of locus
`loc_32_1`	first allele of locus
`loc_32_2`	second allele of locus
`loc_33_1`	first allele of locus
`loc_33_2`	second allele of locus
`loc_34_1`	first allele of locus
`loc_34_2`	second allele of locus
`loc_35_1`	first allele of locus
`loc_35_2`	second allele of locus
`loc_36_1`	first allele of locus
`loc_36_2`	second allele of locus

`loc_37_1`	first allele of locus
`loc_37_2`	second allele of locus
`loc_38_1`	first allele of locus
`loc_38_2`	second allele of locus
`loc_39_1`	first allele of locus
`loc_39_2`	second allele of locus
`loc_40_1`	first allele of locus
`loc_40_2`	second allele of locus
`loc_41_1`	first allele of locus
`loc_41_2`	second allele of locus
`loc_42_1`	first allele of locus
`loc_42_2`	second allele of locus
`loc_43_1`	first allele of locus
`loc_43_2`	second allele of locus
`loc_44_1`	first allele of locus
`loc_44_2`	second allele of locus
`loc_45_1`	first allele of locus
`loc_45_2`	second allele of locus
`loc_46_1`	first allele of locus
`loc_46_2`	second allele of locus
`loc_47_1`	first allele of locus
`loc_47_2`	second allele of locus
`loc_48_1`	first allele of locus
`loc_48_2`	second allele of locus
`loc_49_1`	first allele of locus
`loc_49_2`	second allele of locus
`loc_50_1`	first allele of locus
`loc_50_2`	second allele of locus
`loc_51_1`	first allele of locus
`loc_51_2`	second allele of locus
`loc_52_1`	first allele of locus
`loc_52_2`	second allele of locus
`loc_53_1`	first allele of locus
`loc_53_2`	second allele of locus
`loc_54_1`	first allele of locus
`loc_54_2`	second allele of locus
`loc_55_1`	first allele of locus

`loc_55_2`	second allele of locus
`loc_56_1`	first allele of locus
`loc_56_2`	second allele of locus
`loc_57_1`	first allele of locus
`loc_57_2`	second allele of locus
`loc_58_1`	first allele of locus
`loc_58_2`	second allele of locus
`loc_59_1`	first allele of locus
`loc_59_2`	second allele of locus
`loc_60_1`	first allele of locus
`loc_60_2`	second allele of locus
`loc_61_1`	first allele of locus
`loc_61_2`	second allele of locus
`loc_62_1`	first allele of locus
`loc_62_2`	second allele of locus
`loc_63_1`	first allele of locus
`loc_63_2`	second allele of locus
`loc_64_1`	first allele of locus
`loc_64_2`	second allele of locus
`loc_65_1`	first allele of locus
`loc_65_2`	second allele of locus
`loc_66_1`	first allele of locus
`loc_66_2`	second allele of locus
`loc_67_1`	first allele of locus
`loc_67_2`	second allele of locus
`loc_68_1`	first allele of locus
`loc_68_2`	second allele of locus
`loc_69_1`	first allele of locus
`loc_69_2`	second allele of locus
`loc_70_1`	first allele of locus
`loc_70_2`	second allele of locus
`loc_71_1`	first allele of locus
`loc_71_2`	second allele of locus
`loc_72_1`	first allele of locus
`loc_72_2`	second allele of locus
`loc_73_1`	first allele of locus
`loc_73_2`	second allele of locus

`loc_74_1`	first allele of locus
`loc_74_2`	second allele of locus
`loc_75_1`	first allele of locus
`loc_75_2`	second allele of locus
`loc_76_1`	first allele of locus
`loc_76_2`	second allele of locus
`loc_77_1`	first allele of locus
`loc_77_2`	second allele of locus
`loc_78_1`	first allele of locus
`loc_78_2`	second allele of locus
`loc_79_1`	first allele of locus
`loc_79_2`	second allele of locus
`loc_80_1`	first allele of locus
`loc_80_2`	second allele of locus
`loc_81_1`	first allele of locus
`loc_81_2`	second allele of locus
`loc_82_1`	first allele of locus
`loc_82_2`	second allele of locus
`loc_83_1`	first allele of locus
`loc_83_2`	second allele of locus
`loc_84_1`	first allele of locus
`loc_84_2`	second allele of locus
`loc_85_1`	first allele of locus
`loc_85_2`	second allele of locus
`loc_86_1`	first allele of locus
`loc_86_2`	second allele of locus
`loc_87_1`	first allele of locus
`loc_87_2`	second allele of locus
`loc_88_1`	first allele of locus
`loc_88_2`	second allele of locus
`loc_89_1`	first allele of locus
`loc_89_2`	second allele of locus
`loc_90_1`	first allele of locus
`loc_90_2`	second allele of locus
`loc_91_1`	first allele of locus
`loc_91_2`	second allele of locus
`loc_92_1`	first allele of locus

`loc_92_2`	second allele of locus
`loc_93_1`	first allele of locus
`loc_93_2`	second allele of locus
`loc_94_1`	first allele of locus
`loc_94_2`	second allele of locus
`loc_95_1`	first allele of locus
`loc_95_2`	second allele of locus
`loc_96_1`	first allele of locus
`loc_96_2`	second allele of locus
`loc_97_1`	first allele of locus
`loc_97_2`	second allele of locus
`loc_98_1`	first allele of locus
`loc_98_2`	second allele of locus
`loc_99_1`	first allele of locus
`loc_99_2`	second allele of locus
`loc_100_1`	first allele of locus
`loc_100_2`	second allele of locus

Source

Computed with `python` and provided by S. Stoeckel.

Examples

```
data(popsim)
```

`posidonia` *Posidonia*

Description

A sub-sample table of a large dataset of *Posidonia oceanica* sampled in mediterranean sea.

Usage

```
data("posidonia")
```

Format

A data frame with 40 observations on the following 14 variables.

`Po15_1`	first allele of locus Po15
`Po15_2`	second allele of locus Po15
'`Po4-3_1`'	first allele of locus Po4-3
'`Po4-3_2`'	second allele of locus Po4-3
'`Po5-10_1`'	first allele of locus Po5-10
'`Po5-10_2`'	second allele of locus Po5-10
'`Po5-39_1`'	first allele of locus Po5-39
'`Po5-39_2`'	second allele of locus Po5-39
'`Po5-40_1`'	first allele of locus Po5-40
'`Po5-40_2`'	second allele of locus Po5-40
'`Po5-49_1`'	first allele of locus Po5-49
'`Po5-49_2`'	second allele of locus Po5-49
`Po5_1`	first allele of locus Po5
`Po5_2`	second allele of locus Po5

Source

Arnaud-Haond S, Alberto F, Eguiluz VM, Hernandez-Garcia E, Duarte CM, Serrao EA (2014)

Data from: Disentangling the influence of mutation and migration in clonal seagrasses using the Genetic Distance Spectrum for microsatellites.

Dryad Digital Repository. `http://dx.doi.org/10.5061/dryad.3b8k6`

References

Arnaud-Haond S, Maolic Y, Hernandez-Garcia E, Eguiluz VM, Alberto F, Serrao EA, Duarte CM (2014) Disentangling the influence of mutation and migration in clonal seagrasses using the Genetic Distance Spectrum for microsatellites. Journal of Heredity 105(4): 532-541. `http://dx.doi.org/10.1093/jhered/esu015`

Arnaud-Haond S & Belkhir K, 2007, GENCLONE: a computer program to analyse genotypic data, test for clonality and describe spatial clonal organization.

Examples

```
data(posidonia)
```

psex *Probability of originating from distinct sexual events*

Description

psex and psex_Fis compute the probability that repeated genotypes originate from distinct sexual events (i.e. being different genets and not ramets of the same MLG), with or without taking account of H-W equilibrium departures.

Usage

```
psex(data1, haploid = FALSE, vecpop = NULL, genet = FALSE, RR = FALSE,
MLGsim = FALSE,  nbrepeat = NULL, bar = FALSE)
psex_Fis(data1, vecpop = NULL, genet = FALSE, RR = FALSE, MLGsim = FALSE,
nbrepeat = NULL, bar = FALSE)
```

Arguments

data1	a Rclone table with one allele per column.
haploid	logical, option, haploid indicates the ploidy level of data1. Not edible for psex_Fis.
vecpop	vector, option, vecpop indicates the population name of each unit of data1, if data1 contains several populations. If data1 contains only one population, leave vecpop = NULL.
genet	option, if genet = TRUE, computes pgen on genet level.
RR	option, if RR = TRUE, computes pgen with Round-Robin method.
MLGsim	option, the method of psex calculation (see details).
nbrepeat	option, numeric, the population is simulated nbrepeat times, based on frequency values.
bar	option, if TRUE, a progression bar appears.

Details

We strongly recommand to use RR = TRUE option to compute allelic frequencies for clonal data. Otherwise, we let the options to work with frequencies at genet level (genet = TRUE) or ramet level (RR = FALSE and genet = FALSE).

if MLGsim = TRUE, psex are computed as probability for two units to be derived from distinct sexual reproductive event to be C(N,2) (Stenberg et al. 2003).

If MLGsim = FALSE, psex are computed with more conservative C(n,1) (Parks & Werth 1993) with n, *"number of separated fragments with identical genotype to some previously encountered ramet"*.

The pvalue method calculation is largely inspired from MLGsim (Stenberg et al., 2003) and MLGsim2.0 (Ivens et al., 2012), with authors agreements.
For each repeat, a population is simulated with allelic frequencies.
If clones occurred, a simulated psex is computed and kept in memory.

At the end, a distribution of `sim psex` is constructed and p-value is computed as upper p-value (Monte Carlo).

`psex` and `psex_Fis` could be time consuming with a certain number of repeats.

Values must differ from MLGsim and MLGsim2.0 because of Round-Robin frequencies and `Fis` calculation (see `freq_RR` and `Fis`).

Value

For one population:

- if `nbrepeat` is not provided, a table with psex values,
- if `nbrepeat` is provided, a list of a table with `psex` values and p-values and a vector of `sim psex`.

If `data1` is a multi-population table (`vecpop != NULL`), a list of either tables/tables and vectors for each population.

Warning

If `sim_psex` are less than 100, a warning message pops, as clones are not necessarily generated each simulation.

If no repeated genotype is generated during simulations, a warning message pops as well.

Author(s)

Creator/Author: Diane Bailleul <diane.bailleul.pro@gmail.com>
Author: Sophie Arnaud-Haond <sophie.arnaud@ifremer.fr>
Contributor: Solenn Stoeckel

The R implementation of `RClone` was written by Diane Bailleul.

The design was inspired by GenClone program described in Arnaud-Haond & Belkhir (2007).

References

Stenberg et al., 2003, MLGsim: a program for detecting clones using a simulation approach.

Arnaud-Haond et al., 2007, Standardizing methods to address clonality in population studies.

Ivens, A.B.F., van de Sanden, M. and Bakker, J. MLGsim 2.0: updated software for detecting clones from micro satellite data using a simulation approach.
In: The Evolutionary Ecology of Mutualism. PhD Thesis, 2012, University of Groningen. Pg 107-111
`http://www.rug.nl/research/institute-evolutionary-life-sciences/tres/downloads` for MLGsim 2.0.

See Also

`Fis`, `freq_RR`, `pgen` and `pgen_Fis`

Examples

```
data(posidonia)

psex(posidonia, RR = TRUE)
psex(posidonia, RR = TRUE, MLGsim = TRUE)
#psex(posidonia, RR = TRUE, nbrepeat = 1000, bar = TRUE)
##time consuming
```

`resvigncont` *Results contained in vignette Quick Manual*

Description

This file contains data to fast generate the vignette: RClone_quickmanual.

Usage

```
data("resvigncont")
```

Format

The format is: List of 14 $ resee :List of 2 ..$ results :'data.frame': 1 obs. of 3 variables: .. ..$ Ee : num 0.0779 .. ..$ pval_Ee : num 0.434 .. ..$ nbrepeat: num 1000 ..$ simulations: num [1:1000] 0.0316 -0.1692 -0.1411 -0.1172 -0.1289 ... $ resgen :'data.frame': 1 obs. of 24 variables: ..$ N :List of 1 .. ..$: int 40 ..$ Lineage :List of 1 .. ..$: chr "MLG" ..$ nb_L :List of 1 .. ..$: int 28 ..$ nb_all :List of 1 .. ..$: num 4.14 ..$ SE :List of 1 .. ..$: num 0.769 ..$ Fis :List of 1 .. ..$: num 0.0508 ..$ pval_2sides :List of 1 .. ..$: logi NA ..$ Fis_WR :List of 1 .. ..$: num 0.0257 ..$ pval_2sides :List of 1 .. ..$: logi NA ..$ R :List of 1 .. ..$: num 0.692 ..$ Pareto_index:List of 1 .. ..$: num 1.18 ..$ Sp_Loiselle :List of 1 .. ..$: num 0.00123 ..$ pval_2sides :List of 1 .. ..$: num NA ..$ Sp_L_WR :List of 1 .. ..$: num 0.00124 ..$ pval_2sides :List of 1 .. ..$: num NA ..$ Sp_Ritland :List of 1 .. ..$: num 0.000769 ..$ pval_2sides :List of 1 .. ..$: num NA ..$ Sp_R_WR :List of 1 .. ..$: num 0.000803 ..$ pval_2sides :List of 1 .. ..$: num NA ..$ H" :List of 1 .. ..$: num 3.15 ..$ J' :List of 1 .. ..$: num 0.945 ..$ D :List of 1 .. ..$: num 0.971 ..$ V :List of 1 .. ..$: num 0.792 ..$ Hill :List of 1 .. ..$: num 33.9 $ resagg :List of 2 ..$ results : num [1, 1:3] 0.227 0 1000 .. ..- attr(*, "dimnames")=List of 2 .. .. ..$: NULL .. .. ..$: chr [1:3] "Ac" "pval" "nbrepeat" ..$ simulation: num [1:1000] 0.034 0.034 0.034 0.034 0.034 ... $ rescs :List of 2 ..$: num 11.7 ..$ clonal_sub_tab:List of 60 .. ..$: int 97 .. ..$: int 157 .. ..$: int 119 .. ..$: int 110 .. ..$: int 121 .. ..$: int 64 .. ..$: int 34 .. ..$: int 29 .. ..$: int 31 .. ..$: int 18 .. ..$: num 0.5 .. ..$: num 7.62 .. ..$: num 15.2 .. ..$: num 22.9 .. ..$: num 30.5 .. ..$: num 38 .. ..$: num 46.1 .. ..$: num 53.5 .. ..$: num 61 .. ..$: num 68.5 .. ..$: num 7.52 .. ..$: num 15.2 .. ..$: num 22.8 .. ..$: num 30.4 .. ..$: num 38 .. ..$: num 45.6 .. ..$: num 53.1 .. ..$: num 60.7 .. ..$: num 68 .. ..$: num 76 .. ..$: num 4.68 .. ..$: num 11.1 .. ..$: num 18.8 .. ..$: num 26.6 .. ..$: num 34.2 .. ..$: num 41.5 .. ..$: num 49.6 .. ..$: num 57.1 .. ..$: num 64.7 .. ..$: num 70.9 .. ..$: num 0.165 .. ..$: num 0.0446 .. ..$: num 0 .. ..$: num 0 .. ..$: num 0 .. ..$: num 0 .. ..$: num 0 .. ..$: num 0 .. ..$: num 0 .. ..$: num 0 .. ..$: num -0.783 .. ..$: num -1.35 .. ..$: num -Inf .. ..$: num -Inf .. ..$: num -Inf .. ..$: num -Inf .. ..$: num -Inf .. ..$: num -Inf .. ..$: num -Inf .. ..$: num -Inf .. ..- attr(*, "dim")= int [1:2] 10 6 .. ..- attr(*, "dimnames")=List of 2 .. .. ..$: chr [1:10] "1" "2" "3" "4" ... .$: chr [1:6] "nb_pairs" "dist_min" "dist_max" "dist_mean" ... $ resauto :List of 7 ..$ Main_results :'data.frame': 10 obs. of 7 variables: .. ..$ dist_min : num [1:10] 0.5 7.62 15.24 22.94 30.5 ... .$ dist_max : num [1:10] 7.52 15.21 22.8 30.41 38 ... .$ dist_mean : num [1:10] 4.68 11.15 18.81 26.65 34.21 ... .$ ln(dist_mean): num [1:10] 1.54 2.41 2.93 3.28 3.53 ... .$ nb_pairs : num [1:10] 97 157 119 110 121 64 34 29 31 18 .. ..$ mean_Ritland : num [1:10] 0.0892 0.0296 -0.0224 -0.0532 -0.0736 ... .$ pval_kin : num [1:10] 0 0 0.39 0 0 0 0.144 0.154 0.8 0.106 ..$ Slope_and_Sp_index :'data.frame': 10 obs. of 4 variables: .. ..$ b : num [1:10] -7.01e-04 1.96e-06 2.75e-04 -6.25e-04 4.65e-04 ... .$ b_log : num [1:10] -3.58e-02 3.47e-05 6.30e-03 -1.42e-02 1.01e-02 ... .$ Sp : num [1:10] 7.69e-04 -7.63e-07 2.73e-04 -4.58e-04 6.18e-04 ... .$ Sp_log: num [1:10] 3.93e-02 9.75e-06 6.24e-03 -9.86e-03 1.41e-02 ... ..$ Slope_resample :'data.frame': 1000 obs. of 4 variables: .. ..$ b : num [1:1000] 8.95e-05 9.99e-05 1.64e-04 1.23e-04 1.53e-04 ... .$ b_log : num [1:1000] 0.004679 0.000447 0.000649 0.004408 0.003708 ... .$ Sp : num [1:1000] -8.65e-05 -9.98e-05 -1.65e-04 -1.19e-04 -1.50e-04 ... .$ Sp_log: num [1:1000] -0.004524 -0.000446 -0.000653 -0.004292 -0.003647 ... ..$ Kinship_resample : num [1:1000, 1:10] -0.03434 -0.00168 0.00721 -0.02697 -0.01677 ... .- attr(*, "dimnames")=List of 2 .. .. ..$: NULL .. .. ..$: chr [1:10] "class_1" "class_2" "class_3" "class_4" ... ..$ Matrix_kinship_results:Class 'dist' atomic [1:780] -0.16518 -0.00174 -0.00174 -0.00174 -0.00174 ... .- attr(*, "Labels")= chr [1:40] "1" "2" "3" "4" ... .- attr(*, "Size")= int 40 .. .. ..- attr(*, "call")= language as.dist.default(m = mat_auto) .. .. ..- attr(*, "Diag")= logi FALSE .. .. ..- attr(*, "Upper")= logi FALSE ..$ Class_kinship_results :List of 10 .. ..$: num [1:97] -0.16518 -0.00174 -0.00174 -0.00174 -0.03193 ... .$: num [1:157] -0.00174 -0.00174 -0.02292 -0.08243 -0.15206 ... .$: num [1:119] -0.0455 -0.0921 -0.0455 -0.0455 0.036 ... .$: num [1:110] -0.0455 -0.1553 -0.1433 -0.047 -0.1513 ... .$: num [1:121] -0.143 -0.143 -0.114 0.301 -0.181 ... .$: num [1:64] -0.0918 -0.1236 -0.1236 -0.1024 -0.1236 ... .$: num [1:34] -0.08461 0.10111 0.1034 -0.00678 0.05459 ... .$: num [1:29] -0.00693 -0.00693 -0.00693 -0.00693 -0.00693 ... .$: num [1:31] 0.3935 -0.0646 -0.1289 -0.0656 -0.0825 ... .$: num [1:18] 0.00503 0.26968 0.00414 -0.02171 -0.04514 ... ..$ Class_distance_results:List of 10 .. ..$: num [1:97] 2 5.7 7.11 5.15 4.74 ... .$: num [1:157] 8.06 8.28 9.96 8.08 13.73 ... .$: num [1:119] 15.4 15.9 17.4 18.7 19.6 ... .$: num [1:110] 24.5 26.1 26.9 24.1 24.9 ... .$: num [1:121] 31.3 31.5 34.5 31.8 32.5 ... .$: num [1:64] 38.5 38.1 38.5 39.5 39.5 ... .$: num [1:34] 47.4 47.6 52 50 48.9 ... .$: num [1:29] 60.6 60.7 59.8 58.7 56.5 ... .$: num [1:31] 63.4 66 61.4 64.1 68 ... .$: num [1:18] 70 70.8 70.5 73.1 76 ... $ rescl :'data.frame': 1 obs. of 7 variables: ..$ G : num 28 ..$ R : num 0.692 ..$ H" : num 3.15 ..$ J' : num 0.945 ..$ D : num 0.971 ..$ V : num 0.792 ..$ Hill: num 33.9 $ ressimWS:List of 1 ..$ distance_matrix:Class 'dist' atomic [1:499500] 59 126 71 59 63 120 119 164 105 80 ... .- attr(*, "Labels")= chr [1:1000] "1" "2" "3" "4" ... .- attr(*, "Size")= int 1000 .. .. ..- attr(*, "call")= language as.dist.default(m = dist_all) .. .. ..- attr(*, "Diag")= logi FALSE .. .. ..- attr(*, "Upper")= logi FALSE $ ressim :List of 1 ..$ distance_matrix:Class 'dist' atomic [1:499500] 143 98 138 93 106 98 106 113 109 102 ... .- attr(*, "Labels")= chr [1:1000] "1" "2" "3" "4" ... .- attr(*, "Size")= int 1000 .. .. ..- attr(*, "call")= language as.dist.default(m = dist_all) .. .. ..- attr(*, "Diag")= logi FALSE .. .. ..- attr(*, "Upper")= logi FALSE $ respop :List of 1 ..$ distance_matrix:Class 'dist' atomic [1:153] 173 178 2 173 178 1 1 84 133 100 ... .- attr(*, "Labels")= chr [1:18] "1" "2" "3" "4" ... .- attr(*, "Size")= int 18 .. .. ..- attr(*, "call")= language as.dist.default(m = dist_all) .. .. ..- attr(*, "Diag")= logi FALSE .. .. ..- attr(*, "Upper")= logi FALSE $ res_PS4 :'data.frame': 6 obs. of 4 variables: ..$ pgenFis: num [1:6] 1.05e-05 1.09e-10 4.39e-05 4.39e-05 4.39e-05 ... ..$ genet : chr [1:6] "" "" "" "3" ... ..$ psexFis: chr [1:6] "" "" "" "0.00175402908240928" ... ..$ pvalue : chr [1:6] "" "" "" "0.258064516129032" ... $ res_PS3 :List of 2 ..$:'data.frame': 40 obs. of 3 variables: .. ..$ genet : chr [1:40] "" "" "" "3" ... .$ psexFis: chr [1:40] "" "" "" "0.00175402908240928" ... .$ pvalue : chr [1:40] "" "" "" "0.258064516129032"

... ..$: num [1:31] 0.004048 0.00316 0.009211 0.000587 0.007884 ... $ res_PS2 :'data.frame': 6 obs. of 4 variables: ..$ pgen : num [1:6] 2.19e-06 2.04e-10 4.77e-05 4.77e-05 4.77e-05 ... ..$ genet : chr [1:6] "" "" "" "3" ... ..$ psex : chr [1:6] "" "" "" "0.00190284159898287" ... ..$ pvalue: chr [1:6] "" "" "" "0.392857142857143" ... $ res_PS1 :List of 2 ..$:'data.frame': 40 obs. of 3 variables: .. ..$ genet : chr [1:40] "" "" "" "3" ... .$ psex : chr [1:40] "" "" "" "0.00190284159898287" ... .$ pvalue: chr [1:40] "" "" "" "0.392857142857143" ... ..$: num [1:28] 0.00268 0.00135 0.0034 0.00154 0.0043 ... $ res_SU1 :List of 5 ..$ res_MLG :'data.frame': 7 obs. of 5 variables: .. ..$ nb_loci : int [1:7] 1 2 3 4 5 6 7 .. ..$ min : int [1:7] 3 7 11 19 22 25 28 .. ..$ max : int [1:7] 13 21 26 27 28 28 28 .. ..$ mean_MLG: num [1:7] 6.26 14.27 20.14 23.57 25.44 ... .$ SE : num [1:7] 0.1047 0.1362 0.0966 0.0618 0.046 ... ..$ res_alleles:'data.frame': 7 obs. of 7 variables: .. ..$ nb_loci : int [1:7] 1 2 3 4 5 6 7 .. ..$ min : int [1:7] 2 5 8 11 15 22 29 .. ..$ max : int [1:7] 7 14 18 21 24 27 29 .. ..$ mean_all: num [1:7] 4.09 8.33 12.42 16.53 20.7 ... .$ SE : num [1:7] NA 132.3 88.3 70.2 60.7 ... .$ He : num [1:7] 0.549 0.549 0.55 0.55 0.55 ... .$ SE : num [1:7] NA 1.217 0.803 0.646 0.552 ... ..$ raw_He :'data.frame': 1000 obs. of 7 variables: .. ..$ 1_locus: num [1:1000] 0.531 0.38 0.488 0.488 0.657 ... .$ 2_loci : num [1:1000] 0.621 0.771 0.459 0.519 0.509 ... .$ 3_loci : num [1:1000] 0.483 0.691 0.677 0.558 0.658 ... .$ 4_loci : num [1:1000] 0.517 0.616 0.613 0.559 0.616 ... .$ 5_loci : num [1:1000] 0.538 0.523 0.588 0.517 0.538 ... .$ 6_loci : num [1:1000] 0.534 0.534 0.512 0.6 0.534 ... .$ 7_loci : num [1:1000] 0.551 0.551 0.551 0.551 0.551 ... ..$ raw_MLG :'data.frame': 1000 obs. of 7 variables: .. ..$ 1_locus: int [1:1000] 4 3 6 6 6 6 9 6 6 3 ... .$ 2_loci : int [1:1000] 19 20 12 12 13 12 21 14 17 19 ... .$ 3_loci : int [1:1000] 18 21 23 24 24 19 18 20 18 20 ... .$ 4_loci : int [1:1000] 24 25 21 26 25 23 26 20 21 26 ... .$ 5_loci : int [1:1000] 26 27 24 26 26 25 24 27 26 25 ... .$ 6_loci : int [1:1000] 25 25 26 27 25 27 28 28 28 26 ... .$ 7_loci : int [1:1000] 28 28 28 28 28 28 28 28 28 28 ... ..$ raw_all :'data.frame': 1000 obs. of 7 variables: .. ..$ 1_locus: int [1:1000] 4 2 3 3 3 3 7 3 3 2 ... .$ 2_loci : int [1:1000] 10 14 6 5 7 5 10 6 9 10 ... .$ 3_loci : int [1:1000] 10 18 17 13 14 8 8 9 9 14 ... .$ 4_loci : int [1:1000] 17 17 20 17 17 12 17 16 19 17 ... .$ 5_loci : int [1:1000] 20 19 23 19 20 24 23 24 19 18 ... .$ 6_loci : int [1:1000] 26 26 22 26 26 26 26 27 27 22 ... .$ 7_loci : int [1:1000] 29 29 29 29 29 29 29 29 29 29 ...

Author(s)

Creator/Author: Diane Bailleul <diane.bailleul.pro@gmail.com>
Author: Sophie Arnaud-Haond <sophie.arnaud@ifremer.fr>
Contributor: Solenn Stoeckel

The R implementation of `RClone` was written by Diane Bailleul.

The design was inspired by GenClone program described in Arnaud-Haond & Belkhir (2007).

Examples

```
#v1 <- vignette("RClone_quickmanual")
#print(v1)
```

`sample_LU` *Sample Loci or Units*

Description

Monte Carlo procedure to ensure that the sets of loci (`sample_units`) or units (`sample_loci`) provide enough power to discriminate MLG (MultiLocus Genotypes).

Usage

```
sample_loci(data1, haploid = FALSE, vecpop = NULL, nbrepeat = 1000, He = FALSE,
graph = FALSE, export = FALSE, bar = FALSE)
sample_units(data1, haploid = FALSE, vecpop = NULL, nbrepeat = 1000, He = FALSE,
graph = FALSE, export = FALSE, bar = FALSE)
```

Arguments

`data1`	a `Rclone` table with one allele per column.
`haploid`	logical, option, `haploid` indicates the ploidy level of `data1`. Not edible for `pgen_Fis`.
`vecpop`	vector, option, `vecpop` indicates the population name of each unit of `data1`, if `data1` contains several populations. If `data1` contains only one population, leave `vecpop = NULL`.
`nbrepeat`	numeric, the number of sampling.
`He`	option, if `TRUE`, computes `Hexp` (expected Heterozygosity, Nei 1978).
`graph`	option, if `TRUE`, displays a boxplot of average MLG number using X loci.
`export`	option, if `TRUE`, graph is saved as pdf into working directory.
`bar`	option, if `TRUE`, displays a progression bar.

Value

a list of:

`res_MLG`	with min, max, mean and SE (Standard Error) of MLG,
`res_alleles`	with min, max, mean and Satterthwaite approximation of SE of the number of alleles and of `Hexp` if option `He = TRUE`,
`raw_He`	a table with number of loci/units sampled in column and each re-sampling in row for He,
`raw_MLG`	a table with number of loci/units sampled in column and each re-sampling in row for MLG number,
`raw_all`	a table with number of loci/units sampled in column and each re-sampling in row for alleles number.

If `data1` is a multi-population table, a list of lists for each population.

Author(s)

Creator/Author: Diane Bailleul <diane.bailleul.pro@gmail.com>
Author: Sophie Arnaud-Haond <sophie.arnaud@ifremer.fr>
Contributor: Solenn Stoeckel

The R implementation of `RClone` was written by Diane Bailleul.

The design was inspired by GenClone program described in Arnaud-Haond & Belkhir (2007).

References

Arnaud-Haond et al., 2007, Standardizing methods to address clonality in population studies.

Examples

```
data(posidonia)

sample_loci(posidonia, nbrepeat = 10, graph = TRUE)[[2]]
sample_units(posidonia, nbrepeat = 10, graph = TRUE, bar = TRUE, He = TRUE)[[1]]

#Graph :
res <- sample_loci(posidonia, nbrepeat = 100)
boxplot(res$raw_MLG, range = 3, ylab = "Number of multilocus genotypes",
xlab = "Number of loci sampled")
title(paste("Genotype accumulation curve for", "posidonia"))
```

`sort_all` *Sorting alleles*

Description

`sort_all` sorts alleles of diploid data by increasing order.

Usage

```
sort_all(data1)
```

Arguments

`data1`	a `Rclone` table with one allele per column.

Details

To use properly `RClone` functions on diploid data, you **MUST** be sure that your alleles are sorted by increasing order.
Run this function before any analysis.

Value

a table of exact format of `data1`, but with alleles sorted.

Author(s)

Creator/Author: Diane Bailleul <diane.bailleul.pro@gmail.com>
Author: Sophie Arnaud-Haond <sophie.arnaud@ifremer.fr>
Contributor: Solenn Stoeckel

The R implementation of `RClone` was written by Diane Bailleul.

The design was inspired by GenClone program described in Arnaud-Haond & Belkhir (2007).

References

Arnaud-Haond et al., 2007, Standardizing methods to address clonality in population studies.

See Also

`convert_GC` for tables with one locus per column.

Examples

```
data(posidonia)

posidonia == sort_all(posidonia)
```

`transcript_GC` *Transcript GenClone files*

Description

`transcript_GC` allows conversion from GenClone files to `RClone` files.

Usage

```
transcript_GC(obj, ele, num1, num2, num3)
```

Arguments

`obj`	a `.csv` file from GenClone (`.txt` saved as `.csv`).
`ele`	option, separator element for import.
`num1`	numeric, the number of loci.
`num2`	numeric, the ploidy level. 2 for diploids and 1 for haploids.
`num3`	numeric, the length of each allele.

Details

GenClone files are generally `.txt` files named `infile.txt`. You must save it as `.csv` file with `";"` as separators and, if necessary, change `","` by `"."`.

Value

`transcript_GC` returns a list of:

`data_genet`	a table of genotypes, one allele per column and one unit per row,
`data_coord`	a table of x/y coordinates,
`names_loci`	a vector of names of the loci,
`names_units`	a vector of names of the units.

Note

`transcript_GC` works only with infile files full of informations (loci names, ploidy names, etc.).

Author(s)

Creator/Author: Diane Bailleul <diane.bailleul.pro@gmail.com>
Author: Sophie Arnaud-Haond <sophie.arnaud@ifremer.fr>
Contributor: Solenn Stoeckel

The R implementation of `RClone` was written by Diane Bailleul.

The design was inspired by GenClone program described in Arnaud-Haond & Belkhir (2007).

References

Arnaud-Haond et al., 2007, Standardizing methods to address clonality in population studies.

See Also

`sort_all` for sorting users tables with one allele per column.

Examples

```
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)
#data1 <- res$data_genet
#coord <- res$data_coord
```

`zostera` *Zostera Dataset*

Description

A sub-sample table of a large dataset of *Zostera marina* sampled in Brittany, France.

Usage

```
data("zostera")
```

Format

A data frame with 59 observations on the following 12 variables.

`population` a character vector indicating the population

`x` a character vector indicating the population

`y` a character vector indicating the population

`GA12` first locus

`GA16` second locus

`GA17D` third locus

`GA17H` fourth locus

`GA19` fifth locus

`GA2` sixth locus

`GA20` seventh locus

`GA23` eighth locus

`GA35` ninth allele of locus

Source

Becheler R, Benkara E, Moalic Y, Hily C, Arnaud-Haond S (2013)

Data from: Scaling of processes shaping the clonal dynamics and genetic mosaic of seagrasses through temporal genetic monitoring.

Dryad Digital Repository. `http://dx.doi.org/10.5061/dryad.1vp70`

References

Becheler R, Benkara E, Moalic Y, Hily C, Arnaud-Haond S (2013) Scaling of processes shaping the clonal dynamics and genetic mosaic of seagrasses through temporal genetic monitoring. Heredity 112(2): 114-121.

`http://dx.doi.org/10.1038/hdy.2013.82`

Examples

```
data(zostera)
popvec <- zostera[,1]
coord_zostera <- zostera[,2:3]
zostera <- convert_GC(zostera[,4:ncol(zostera)], 3)
```

Index