Package 'pegas'

August 21, 2014

Version 0.6

Date 2014-08-21

Title Population and Evolutionary Genetics Analysis System

Depends R (>= 2.6.0), ape (>= 2.4), adegenet

Imports graphics, utils

ZipData no

Description pegas provides functions for reading, writing, plotting,analysing, and manipulating allelic and haplotypic data, and for the
analysis of population nucleotide sequences and micro-satellites including coalescence analyses.

License GPL (>= 2)

URL http://ape-package.ird.fr/pegas.html

Author Emmanuel Paradis [aut, cre, cph],Klaus Schliep [aut, cph],Alastair Potts [aut, cph],David Winter [aut, cph]

Maintainer Emmanuel Paradis <Emmanuel.Paradis@ird.fr>

NeedsCompilation no

Repository CRAN

Date/Publication 2014-08-21 10:24:22

R topics documented:

`pegas-package` *Population and Evolutionary Genetics Analysis System*

Description

pegas provides functions for the analysis of allelic data and of haplotype data from DNA sequences. It requires and complements two other R-packages: **ape** and **adegenet**.

The complete list of functions can be displayed with `library(help = pegas)`.

More information on **pegas** can be found at `http://ape-package.ird.fr/pegas/`.

Author(s)

Emmanuel Paradis, Alastair Potts, Klaus Schliep, David Winter

Maintainer: Emmanuel Paradis

amova *Analysis of Molecular Variance*

Description

This function performs a hierarchical analysis of molecular variance as described in Excoffier et al. (1992). This implementation accepts any number of hierarchical levels.

Usage

```
amova(formula, data = NULL, nperm = 1000, is.squared = FALSE)
## S3 method for class 'amova'
print(x, ...)
```

Arguments

formula	a formula giving the AMOVA model to be fitted with the distance matrix on the left-hand side of the ~, and the population, region, etc, levels on its right-hand side (see details).
data	an optional data frame where to find the hierarchical levels; by default they are searched for in the user's workspace.
nperm	the number of permutations for the tests of hypotheses (1000 by default). Set this argument to 0 to skip the tests and simply estimate the variance components.
is.squared	a logical specifying whether the distance matrix has already been squared.
x	an object of class "amova".
...	unused (here for compatibility.

Details

The formula must be of the form `d ~ A/B/...` where `d` is a distance object, and `A`, `B`, etc, are the hierarchical levels from the highest to the lowest one. Any number of levels is accepted, so specifying `d ~ A` will simply test for population differentiation.

It is assumed that the rows of the distance matrix are in the same order than the hierarchical levels (which may be checked by the user).

Value

An object of class "amova" which is a list with a table of sums of square deviations (SSD), mean square deviations (MSD), and the number of degrees of freedom, and a vector of variance components.

Note

If there are more than three levels, approximate formulae are used to estimate the variance components.

If there is an error message like this:

```
Error in FUN(X[[1L]], ...) : 'bin' must be numeric or a factor
```

it may be that the factors you use in the formula were not read correctly. You may convert them with the function `factor`, or, before reading your data files, do this command (in case this option was modified):

```
options(stringsAsFactors = TRUE)
```

Author(s)

Emmanuel Paradis

References

Excoffier, L., Smouse, P. E. and Quattro, J. M. (1992) Analysis of molecular variance inferred from metric distances among DNA haplotypes: application to human mitochondrial DNA restriction data. *Genetics*, **131**, 479–491.

See Also

`amova` in **ade4** for an implementation of the original Excoffier et al.'s model; `adonis` in **vegan** for a general (multivariate) implementation of an ANOVA framework with distances.

Examples

```
### All examples below have 'nperm = 100' for faster execution times.
### The default 'nperm = 1000' is recommended.
require(ape)
data(woodmouse)
d <- dist.dna(woodmouse)
g <- factor(c(rep("A", 7), rep("B", 8)))
p <- factor(c(rep(1, 3), rep(2, 4), rep(3, 4), rep(4, 4)))
amova(d ~ g/p, nperm = 100) # 2 levels
amova(d ~ p, nperm = 100) # 1 level
amova(d ~ g, nperm = 100)

## 3 levels (quite slow):
## Not run:
pop <- gl(64, 5, labels = paste("pop", 1:64))
region <- gl(16, 20, labels = paste("region", 1:16))
conti <- gl(4, 80, labels = paste("conti", 1:4))
dd <- as.dist(matrix(runif(320^2), 320))
amova(dd ~ conti/region/pop, nperm = 100)

## End(Not run)
```

`as.loci` *Conversion Among Allelic Data Classes*

Description

These functions do conversion among different allelic data classes.

Usage

```
as.loci(x, ...)
## S3 method for class 'genind'
as.loci(x, ...)
genind2loci(x)
## S3 method for class 'data.frame'
as.loci(x, allele.sep = "/|", col.pop = NULL, col.loci = NULL, ...)
loci2genind(x)
## S3 method for class 'factor'
as.loci(x, allele.sep = "/|", ...)
## S3 method for class 'character'
as.loci(x, allele.sep = "/|", ...)
```

Arguments

`x`	an object of class `"loci"` or `"genind"`, a data frame, a factor, or a vector of mode character.
`allele.sep`	the character(s) separating the alleles for each locus in the data file (a forward slash by default).
`col.pop`	specifies whether one of the column of the data file identifies the population; default `NULL`, otherwise an integer or a character giving the number or the name of the column.
`col.loci`	a vector of integers or of characters specifying the indices or the names of the columns that are loci. By default, all columns are taken as loci except the one labelled "population", if present or specified.
`...`	further arguments to be passed to or from other methods.

Details

The main objectives of these functions is to provide easy conversion between the data structures of **adegenet** and **pegas**, so both packages can be used together smoothly. In addition, it is possible to create a `"loci"` object directly from a data frame, a vector, or a factor.

`genind2loci(x)` and `as.loci(x)` are the same if `x` is of class `"genind"`.

Value

An object of class `c("loci", "data.frame")` for `as.loci` and `genind2loci`; an object of class `"genind"` for `loci2genind`.

Author(s)

Emmanuel Paradis

See Also

`read.loci`, `genind`, `df2genind` for converting data frames to `"genind"`

Examples

```
x <- c("A-A", "A-a", "a-a")
as.loci(x, allele.sep = "-")
require(adegenet)
data(nancycats)
x <- as.loci(nancycats)
y <- loci2genind(x) # back to "genind"
identical(nancycats@tab, y@tab)
identical(nancycats@pop, y@pop)
```

`bind.loci` *Bind Loci Objects*

Description

These functions combine objects of class `"loci"` by binding their rows or their columns.

Usage

```
## S3 method for class 'loci'
rbind(...)
## S3 method for class 'loci'
cbind(...)
```

Arguments

`...`	some object(s) of class `"loci"`, separated with commas.

Details

These two methods call `[rc]bind.data.frame` and take care to respect the attribute "locicol" of the returned object.

You can pass a data frame in the `...`, but then you should bypass the generic by calling `cbind.loci` directly. Do not try to pass a vector: this will mess the "locicol" attribute. Instead, make a data frame with this vector (see examples).

Value

An object of class `"loci"`.

Author(s)

Emmanuel Paradis

See Also

[.loci

Examples

```
a <- as.loci(data.frame(x = "A/a", y = 1), col.loci = 1)
b <- as.loci(data.frame(y = 2, x = "A/A"), col.loci = 2)
## rbind.loci reorders the columns if necessary:
str(rbind(a, b))
## cbind sets "locicol" correctly:
str(cbind(a, b))
str(cbind(b, a))
## Unexpected result...
str(cbind(a, data.frame(z = 10)))
## ... bypass the generic:
str(pegas:::cbind.loci(a, data.frame(z = 10)))
## ... or much better: a$z <- 10
## Here "locicol" is not correct...
str(pegas:::cbind.loci(z = 10, a))
## ... instead
str(pegas:::cbind.loci(data.frame(z = 10), a))
```

---

edit.loci *Edit Allelic Data with R's Data Editor*

Description

This allows to edit a data frame of class `"loci"` with R's spreadsheet-like data editor.

Usage

```
## S3 method for class 'loci'
edit(name, edit.row.names = TRUE, ...)
```

Arguments

`name`	an object of class `"loci"`.
`edit.row.names`	a logical specifying to allow editing the rownames, `TRUE` by default (by contrast to data frames).
`...`	further arguments to be passed to or from other methods.

Details

This 'method' of the generic `edit` respects the class and the attribute `"locicol"` of the allelic data frame.

Value

A data frame with class `c("loci", "data.frame")`.

Author(s)

Emmanuel Paradis

See Also

`read.loci`, `summary.loci`

`Fst` *F-Statistics*

Description

This function computes the F_{IT}, F_{ST} and F_{IS} for each locus in the data.

Usage

```
Fst(x, pop = NULL)
```

Arguments

`x`	an object of class `"loci"`.
`pop`	a vector or factor giving the population assignment of each row of `x`, or a single numeric value specifying which column of `x` to use as population indicator. By default, the column labelled `"population"` is used.

Details

The formulae in Weir and Cockerham (1984) are used for each allele, and then averaged within each locus over the different alleles as suggested by these authors.

Value

A matrix with genes (loci) as rows and the three *F*-statistics as columns.

Note

Programming bugs have been fixed in version 0.3-2 of **pegas**. Further tests and feed-back are still welcome.

Author(s)

Emmanuel Paradis

References

Weir, B. S. and Cockerham, C. C. (1984) Estimating F-statistics for the analysis of population structure. *Evolution*, **38**, 1358–1370.

Weir, B. S. and Hill, W. G. (2002) Estimating F-statistics. *Annu Review of Genetics*, **36**, 721–750.

See Also

`fstat` in **adegenet**; package **dirmult** on CRAN that implements various estimators of the Dirichlet-multinomial distribution, including maximum likekihood and the moments estimator of Weir and Hill (2002); `Fst` in **Biodem** that caculates F_{ST} from a "kinship matrix".

Examples

```
require(adegenet)
data(nancycats)
x <- as.loci(nancycats)
Fst(x)
```

`haploFreq` *Haplotype Frequencies With a Covariate*

Description

This utility function extracts the absolute frequencies of haplotypes with respect to a categorical variable (a factor). The output is useful when ploting haplotype networks.

Usage

```
haploFreq(x, fac, split = "_", what = 2, haplo = NULL)
```

Arguments

`x`	a set of DNA sequences (as an object of class `"DNAbin"`).
`fac`	a factor giving the categorical variable (can be missing).
`split`	a single character (see details).
`what`	a single integer (see details).
`haplo`	an object of class `"haplotype"`.

Details

The frequencies of each haplotype in `x` are counted with respect to a factor which is either specified with `fac`, or extracted from the labels of `x`. In the second case, these labels are split with respect to the character specified in `split` and the `what`'th substrings are extracted and taken as the categorical variable (see example).

If `haplo` is specified, the haplotype frequencies are taken from it, otherwise they are calculated from `x`.

Value

a matrix of counts.

Author(s)

Klaus Schliep and Emmanuel Paradis

See Also

haplotype, haploNet

Examples

```
## generate some artificial data from 'woodmouse':
data(woodmouse)
x <- woodmouse[sample(15, size = 50, replace = TRUE), ]
## labels IdXXX_PopXXX_LocXXX
rownames(x) <- paste("Id", 1:50, "_Pop", 1:2, "_Loc", 1:5, sep = "")
head(labels(x))
h <- haplotype(x)
## frequencies of haplotypes wrt 'Pop':
f.pop <- haploFreq(x, haplo = h)
## frequencies of haplotypes wrt 'Loc':
f.loc <- haploFreq(x, what = 3, haplo = h)
nt <- haploNet(h)
fq <- attr(nt, "freq")
op <- par(mfcol = c(1, 2))
plot(nt, size = fq, pie = f.pop, labels = FALSE)
plot(nt, size = fq, pie = f.loc, labels = FALSE)
par(op)
```

---

haploNet *Haplotype Networks*

Description

haploNet computes a haplotype network. There is a plot method and two conversion functions towards other packages.

Usage

```
haploNet(h, d = NULL)
## S3 method for class 'haploNet'
plot(x, size = 1, col = "black", bg = "white",
     col.link = "black", lwd = 1, lty = 1, pie = NULL,
     labels = TRUE, font = 2, cex = 1, scale.ratio = 1,
     asp = 1, legend = FALSE, fast = FALSE, show.mutation = TRUE, ...)
## S3 method for class 'haploNet'
```

```
as.network(x, directed = FALSE, ...)
## S3 method for class 'haploNet'
as.igraph(x, directed = FALSE, use.labels = TRUE, ...)
```

Arguments

h	an object of class "haplotype".
d	an object giving the distances among haplotypes (see details).
x	an object of class "haploNet".
size	a numeric vector giving the diameter of the circles representing the haplotypes: this is in the same unit than the links and eventually recycled.
col	a character vector specifying the colours of the circles; eventually recycled.
bg	a character vector specifying either the colours of the background of the circles (if pie = NULL), or the colours of the slices of the pies; eventually recycled.
col.link	a character vector specifying the colours of the links; eventually recycled.
lwd	a numeric vector giving the width of the links; eventually recycled.
lty	idem for the line types.
pie	a matrix used to draw pie charts for each haplotype; its number of rows must be equal to the number of haplotypes.
labels	a logical specifying whether to identify the haplotypes with their labels (the default).
font	the font used for these labels (bold by default); must be an integer between 1 and 4.
cex	a numerical specifying the character expansion of the labels.
scale.ratio	the ratio of the scale of the links representing the number of steps on the scale of the circles representing the haplotypes. It may be needed to give a value greater than one to avoid overlapping circles.
asp	the aspect ratio of the plot. Do not change the default unless you want to distort your network.
legend	a logical specifying whether to draw the legend, or a vector of length two giving the coordinates where to draw the legend; FALSE by default. If TRUE, the user is asked to click where to draw the legend.
fast	a logical specifying whether to optimize the spacing of the circles; FALSE by default.
show.mutation	a logical specifying whether to display the mutations on the links of the network.
directed	a logical specifying whether the network is directed (FALSE by default).
use.labels	a logical specifying whether to use the original labels in the returned network.
...	further arguments passed to plot.

Details

By default, the haplotype network is built using an infinite site model (i.e., uncorrected or Hamming distance). Users may specify their own distance with the argument d. Currently, there is no check of labels, so the user must make sure that the distances are ordered in the same way than the haplotypes.

Value

haploNet returns an object of class "haploNet" which is a matrix where each row represents a link in the network, the first and second columns give the numbers of the linked haplotypes, the third column, named "step", gives the number of steps in this link, and the fourth column, named "Prob", gives the probability of a parsimonious link as given by Templeton et al. (1992). There are two additional attributes: "freq", the absolute frequencies of each haplotype, and "labels", their labels.

as.network and as.igraph return objects of the appropriate class.

Author(s)

Emmanuel Paradis, Klaus Schliep

References

Templeton, A. R., Crandall, K. A. and Sing, C. F. (1992) A cladistic analysis of phenotypic association with haplotypes inferred from restriction endonuclease mapping and DNA sequence data. III. Cladogram estimation. *Genetics*, **132**, 619–635.

See Also

haplotype, haploFreq, mst

Examples

```
## generate some artificial data from 'woodmouse':
data(woodmouse)
x <- woodmouse[sample(15, size = 110, replace = TRUE), ]
h <- haplotype(x)
(net <- haploNet(h))
plot(net)
## symbol sizes equal to haplotype sizes:
plot(net, size = attr(net, "freq"), fast = TRUE)
plot(net, size = attr(net, "freq"))
plot(net, size=attr(net, "freq"), scale.ratio = 2, cex = 0.8)
```

---

haplotype *Haplotype Extraction and Frequencies*

Description

haplotype extracts the haplotypes from a set of DNA sequences. The result can be plotted with the appropriate function.

Usage

```
haplotype(x, ...)
## S3 method for class 'DNAbin'
haplotype(x, labels = NULL, ...)
## S3 method for class 'haplotype'
plot(x, ...)
## S3 method for class 'haplotype'
print(x, ...)
## S3 method for class 'haplotype'
sort(x,
     decreasing = ifelse(what == "frequencies", TRUE, FALSE),
     what = "frequencies", ...)
## S3 method for class 'haplotype'
x[...]
```

Arguments

x	a set of DNA sequences (as an object of class "DNAbin"), or an object of class "haplotype".
labels	a vector of character strings used as names for the rows of the returned object. By default, Roman numerals are given.
...	further arguments passed to barplot (unused in print and sort).
decreasing	a logical value specifying in which order to sort the haplotypes; by default this depends on the value of what.
what	a character specifying on what feature the haplotypes should be sorted: this must be "frequencies" or "labels", or an unambiguous abbreviation of these.

Details

The sort method sorts the haplotypes in decreasing frequencies (the default) or in alphabetical order of their labels (if what = "labels"). Note that if these labels are Roman numerals (as assigned by haplotype), their alphabetical order may not be their numerical one (e.g., IX is alphabetically before VIII).

Value

haplotype returns an object of class c("haplotype", "DNAbin") which is an object of class "DNAbin" with two additional attributes: "index" identifying the index of each observation that share the same haplotype, and "from" giving the name of the original data.

sort returns an object of the same class respecting its attributes.

Author(s)

Emmanuel Paradis

See Also

haploNet, haploFreq, DNAbin for manipulation of DNA sequences in R.

The haplotype method for objects of class "loci" is documented separately: haplotype.loci.

Examples

```
## generate some artificial data from 'woodmouse':
data(woodmouse)
x <- woodmouse[sample(15, size = 110, replace = TRUE), ]
(h <- haplotype(x))
## the indices of the individuals belonging to the 1st haplotype:
attr(h, "index")[[1]]
plot(sort(h))
```

---

haplotype.loci *Haplotype Extraction and Frequencies From Allelic Data*

Description

This function extracts haplotypes from phased genotypes.

Usage

```
## S3 method for class 'loci'
haplotype(x, locus = 1:2, ...)
## S3 method for class 'haplotype.loci'
plot(x, ...)
dist.haplotype.loci(x)
```

Arguments

x	an object of class "loci" or of class "haplotype.loci".
locus	a vector of integers giving the loci to analyse.
...	arguments passed to and from methods.

Details

The individuals with at least one unphased genotype are ignored with a warning.

dist.haplotype.loci computes pairwise distances among haplotypes by counting the number of different alleles.

Value

haplotype returns a matrix of mode character with the loci as rows and the haplotypes as columns. The attribute "freq" gives the counts of each haplotype and the class is "haplotype.loci".

dist.haplotype.loci returns an object of class "dist".

Note

haplotype is a generic function with methods for objects of class "DNAbin" and of class "loci". Note that the class returned by these methods is different: c("haplotype", "DNAbin") and "haplotype.loci", respectively. This and other details are likely to change in the future.

Author(s)

Emmanuel Paradis

See Also

haplotype, LD

---

heterozygosity *Heterozygosity at a Locus Using Gene Frequencies*

Description

This function computes the mean heterozygosity from gene frequencies, and returns optionally the associated variance.

Usage

```
heterozygosity(x, variance = FALSE)
H(x, variance = FALSE)
```

Arguments

x	a vector or a factor.
variance	a logical indicating whether the variance of the estimated heterozygosity should be returned (TRUE), the default being FALSE.

Details

The argument x can be either a factor or a vector. If it is a factor, then it is taken to give the individual alleles in the population. If it is a numeric vector, then its values are taken to be the numbers of each allele in the population. If it is a non-numeric vector, it is a coerced as a factor.

The mean heterozygosity is estimated with:

$$\hat{H} = \frac{n}{n-1}\left(1 - \sum_{i=1}^{k} p_i^2\right)$$

where n is the number of genes in the sample, k is the number of alleles, and p_i is the observed (relative) frequency of the ith allele.

Value

A numeric vector of length one with the estimated mean heterozygosity (the default), or of length two if the variance is returned.

Author(s)

Emmanuel Paradis

References

Nei, M. (1987) *Molecular evolutionary genetics*. New York: Columbia University Press.

See Also

theta.s

Examples

```
require(adegenet)
data(nancycats)
## convert the data and compute frequencies:
S <- summary(as.loci(nancycats))
## compute H for all loci:
sapply(S, function(x) H(x$allele))
## ... and its variance
sapply(S, function(x) H(x$allele, variance = TRUE))
```

---

hw.test *Test of Hardy–Weinberg Equilibrium*

Description

This function tests, for a series of loci, the hypothesis that genotype frequencies follow the Hardy–Weinberg equilibrium.

Usage

```
hw.test(x, B = 1000)
```

Arguments

x	an object of class "loci".
B	the number of replicates for the Monte Carlo procedure.

Details

This test can be performed with any level of ploidy. Two versions of the test are available: the classical χ^2-test based on the expected genotype frequencies calculated from the allelic frequencies, and an exact test based on Monte Carlo permutations of alleles. For the moment, the latter version is available only for diploids. Set `B = 0` if you want to skip the second test.

Value

A matrix with three or four columns with the χ^2-value, the number of degrees of freedom, the associated *P*-value, and possibly the *P*-value from the Monte Carlo test. The rows of this matrix are the different loci in `x`.

Author(s)

Emmanuel Paradis

Examples

```
require(adegenet)
data(nancycats)
x <- as.loci(nancycats)
hw.test(x)
```

LD *Linkage Disequilibrium*

Description

These two functions analyse linkage disequilibrium in the case of phased (`LD`) and unphased (`LD2`) genotypes.

Usage

```
LD(x, locus = 1:2, details = TRUE)
LD2(x, locus = 1:2, details = TRUE)
```

Arguments

`x`	an object of class `"loci"`.
`locus`	a vector of integers giving the two loci to analyse.
`details`	a logical value indicating whether to print the correlation matrix among alleles.

Details

These functions consider a pair of loci and compute the correlations among pairs of alleles.

`LD` first scans the data for unphased genotypes: all individuals with at least one unphased genotype are dropped with a warning. It is based on the observed frequencies of haplotypes (Zaykin et al. 2008). `LD2` is based on the observed frequencies of different genotypes (Schaid 2004).

Both functions accept any number of alleles. `LD` can work with any level of ploidy; `LD2` works with diploid data.

The present version does not test the significance of the T_2 test (Zaykin et al. 2008) with permutations. These authors present simulation results suggesting that the chi-squared approximation has similar type I error rates and power than the test based on permutations even for small sample sizes. Furthermore, this test has better statistical properties than alternatives such as those reported here (LRT and Pearson's test).

Value

For both functions, if `details = FALSE`, only the T2 test is returned.

For `LD`: if `details = TRUE`, a named list with the following elements:

`Observed frequencies`
: the counts of haplotypes in the data.

`Expected frequencies`
: the expected frequencies of haplotypes computed from the observed proportions of alleles under the assumption of no linkage disequilibrium.

`Correlations among alleles`
: the observed correlations among alleles from both loci.

`LRT (G-squared)`
: the likelihood-ratio test of the null hypothesis of no linkage disequilibrium.

`Pearson's test (chi-squared)`
: the chi-squared test based on haplotypes counts.

`T2`
: the T_2 test with its number of degrees of freedom (df).

For `LD2`: if `details = TRUE`, a named list with two elements:

`Delta`
: the correlations among alleles (denoted $Delta$ in Schaid 2004).

`T2`
: the T_2 test with its number of degrees of freedom (df).

Author(s)

Emmanuel Paradis

References

Schaid, D. J. (2004) Linkage disequilibrium testing when linkage phase is unknown. *Genetics*, **166**, 505–512.

Zaykin, D. V., Pudovkin, A. and Weir, B. S. (2008) Correlation-based inference for linkage disequilibrium with multiple alleles. *Genetics*, **180**, 533–545.

See Also

haplotype.loci, is.phased

Examples

```
require(adegenet)
data(rupica)
z <- as.loci(rupica)
LD2(z, 8:9)
```

MMD *Mismatch Distribution*

Description

This function draws a histogram of the frequencies of pairwise distances from a set of DNA sequences.

Usage

```
MMD(x, xlab = "Distance", main = "", rug = TRUE, legend = TRUE,
    lcol = c("blue", "red"), lty = c(1, 1), ...)
```

Arguments

x	a set of DNA sequences (object of class "DNAbin").
xlab	the label for the x-axis.
main	the title (none by default).
rug	a logical specifying whether to add a rug of the pairwise distances on the horizontal axis (see rug).
legend	a logical specifying whether to draw a legend.
lcol	the colours used for the curves.
lty	the line types for the curves
...	further arguments passed to hist.

Details

The histogram shows the observed distribution of pairwise distances. The lines show an empirical density estimate (in blue) and the expected distribution under stable population population (Rogers and Harpending 1992).

Author(s)

Emmanuel Paradis and David Winter

References

Rogers, A. R. and Harpending, H. (1992) Population growth makes waves in the distribution of pairwise genetic-differences. *Molecular Biology and Evolution*, **9**, 552–569.

Examples

```
data(woodmouse)
MMD(woodmouse, col = "grey")
MMD(woodmouse, breaks = 20, legend = FALSE)
MMD(woodmouse, lty = 1:2, lcol = rep("black", 2), col = "lightgrey")
```

`mst` *Minimum Spanning Tree*

Description

This function computes a minimum spanning tree using Kruskal's algorithm.

Usage

```
mst(d)
```

Arguments

`d`	a distance matrix, either as an object of class `"dist"`, or a (square symmetric) matrix.

Value

an object of class `"haploNet"`.

Author(s)

Emmanuel Paradis

References

Kruskal, J. B., Jr. (1956) On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, **7**, 48–50.

See Also

`haploNet`

Examples

```
data(woodmouse)
d <- dist.dna(woodmouse, "n")
(r <- mst(d))
plot(r)
```

nuc.div *Nucleotide Diversity*

Description

This function computes the nucleotide diversity from a sample of DNA sequences.

Usage

```
nuc.div(x, variance = FALSE, pairwise.deletion = FALSE)
```

Arguments

x	a matrix or a list which contains the DNA sequences.
variance	a logical indicating whether to compute the variance of the estimated nucleotide diversity.
pairwise.deletion	a logical indicating whether to delete the sites with missing data in a pairwise way. The default is to delete the sites with at least one missing data for all sequences.

Details

The nucleotide diversity is the sum of the number of differences between pairs of sequences divided by the number of comparisons (i.e. $n(n-1)/2$, where n is the number of sequences).

The variance of the estimated diversity uses formula (10.9) from Nei (1987). This applies only if all sequences are of the same lengths, and cannot be used if `pairwise.deletion = TRUE`. A bootstrap estimate may be in order if you insist on using the latter option.

Value

A numeric vector with one or two values if `variance = TRUE`.

Author(s)

Emmanuel Paradis

References

Nei, M. (1987) *Molecular evolutionary genetics*. New York: Columbia University Press.

See Also

base.freq, GC.content, theta.s, seg.sites

Examples

```
data(woodmouse)
nuc.div(woodmouse)
nuc.div(woodmouse, TRUE)
nuc.div(woodmouse, FALSE, TRUE)
```

`R2.test` *Ramos-Onsins–Rozas Test of Neutrality*

Description

This function computes Ramos-Onsins and Rozas's test of neutrality for a set of DNA sequences.

Usage

```
R2.test(x, B = 1000, theta = 1, plot = TRUE, quiet = FALSE, ...)
```

Arguments

`x`	a DNA matrix (object of class `"DNAbin"`).
`B`	the number of replicates used for the simulation procedure.
`theta`	the value of the θ population parameter used in the simulation.
`plot`	a logical value specifying whether to plot the results (`TRUE` by default).
`quiet`	a logical value specifying whether to not display the progress of the simulations. The default is `FALSE` meaning that a progress bar is displayed by default.
`...`	further arguments passed to `hist`.

Value

a list with two elements: `R2` the value of the test statistic R_2, and `P.val` the associated *P*-value. If `B = 0` a single value, the test statistic, is returned

Note

The simulation procedure probably needs to be tested and improved. However the results make sense so far.

Author(s)

Emmanuel Paradis

References

Ramos-Onsins, R. and Rozas, R. (2002) Statistical properties of new neutrality tests against population growth. *Molecular Biology and Evolution*, **19**, 2092–2100.

Sano, J.and Tachida, G. (2005) Gene genealogy and properties of test statistics of neutrality under population growth. *Genetics*, **169**, 1687–1697.

See Also

read.dna, dist.dna

Examples

```
data(woodmouse)
R2.test(woodmouse, quiet = TRUE)
```

---

read.gtx *Read Genetix Data Files*

Description

This function reads allelic data from a Genetix file (.gtx).

Usage

```
read.gtx(file)
```

Arguments

file	a file name specified by either a variable of mode character or a quoted string.

Value

A data frame with class c("loci", "data.frame").

Note

The package **adegenet** has a similar function, read.genetix, but it returns an object of class "genind".

Author(s)

Emmanuel Paradis

References

Belkhir, K., Borsa, P., Chikhi, L., Raufaste, N. and Bonhomme, F. (1996–2004) GENETIX 4.05, logiciel sous Windows(TM) pour la genetique des populations. Laboratoire Genome, Populations, Interactions, CNRS UMR 5000, Universite de Montpellier II, Montpellier (France). http://www.genetix.univ-montp2.fr/genetix/intro.htm

See Also

read.loci, write.loci, read.vcf, read.genetix

Examples

```
require(adegenet)
(X <- read.gtx(system.file("files/nancycats.gtx", package = "adegenet")))
## compare with the example in ?read.genetix
```

`read.loci` *Read Allelic Data Files*

Description

This function reads allelic data from a text file: rows are individuals, and columns are loci and optional variables. By default, the first line of the file gives the locus names. If one column is labelled 'population', it is taken as a population variable.

Usage

```
read.loci(file, header = TRUE, loci.sep = "", allele.sep = "/|",
          col.pop = NULL, col.loci = NULL, ...)
```

Arguments

`file`	a file name specified by either a variable of mode character, or a quoted string.
`header`	a logical specifying whether the first line of the data file gives the names of the loci (`TRUE` by default).
`loci.sep`	the character(s) separating the loci (columns) in the data file (a white space by default).
`allele.sep`	the character(s) separating the alleles for each locus in the data file (a forward slash by default).
`col.pop`	specifies whether one of the column of the data file identifies the population. By default, if one column is labelled 'population' (case-insensitive), it is taken as the population variable; otherwise an integer giving the number of the column or a character string giving its name. It is eventually renamed 'population' and transformed as a factor.
`col.loci`	a vector of integers or characters specifying the indices or the names of the columns that are loci. By default, all columns are taken as loci except the population one, if present or specified.
`...`	further arguments passed to `read.table` (e.g., `row.names`).

Details

The rownames of the returned object identify the individual genotypes; they are either taken from the data file if present, or given the values `"1"`, `"2"`, ... Similarly for the colnames: if absent in the file (in which case `header = FALSE` must be set), they are given the values `"V1"`, `"V2"`, ...

In the returned genotypes, alleles are separated by `"/"`, even if it is not the case in the data file.

The vignette "Reading Genetic Data Files Into R with **adegenet** and **pegas**" explains how to read various file formats including Excel files (type `vignette("ReadingFiles")` in R).

Value

A data frame with class `c("loci", "data.frame")`. It is a data frame with an attribute `"locicol"` specifying the columns that must be treated as loci. The latter are factors. The other columns can be of any type.

Details on the structure can be found in http://ape.mpl.ird.fr/pegas/DefinitionDataClassesPegas.pdf

Author(s)

Emmanuel Paradis

See Also

`read.gtx`, `read.vcf`, `write.loci`, `summary.loci`

---

read.vcf *Read Variant Calling Format Files*

Description

This function reads allelic data from VCF (variant calling format) files.

Usage

```
read.vcf(file, nloci = 1000, skip = 0)
```

Arguments

`file`	a file name specified by either a variable of mode character, or a quoted string.
`nloci`	the number of loci to read.
`skip`	the number of loci to be skipped before reading the genetic data.

Details

The VCF file can compressed (*.gz) or not.

A TABIX file is not required.

Because VCF files can be very big, only 1000 loci are read by default, but 5000 loci can be read rather quickly.

In the VCF standard, missing data are represented by a dot and these are read "as is" by the present function without trying to substitute by `NA`.

Value

an object of class `c("loci", "data.frame")` with four additional attributes:

- CHRthe chromosome number (integer values);
- POSthe position of the locus (numeric values);
- QUALthe quality of the inferred locus (integer values);
- FILTERthe filter (characters).

Author(s)

Emmanuel Paradis

References

http://www.1000genomes.org/node/101

See Also

`read.loci`, `read.gtx`, `write.loci`

---

rr.test *Tajima Relative Rate Test of Molecular Clock*

Description

This function tests the hypothesis of a molecular evolutionary clock (i.e., a constant rate of molecular evolution) between two samples using an outgroup sample. It can be applied to both nucleotide and amino acid sequences.

Usage

```
rr.test(x, y, out)
```

Arguments

`x, y`	a single DNA sequence (object class `"DNAbin"`).
`out`	a single DNA sequence to be used as outgroup.

Value

a list with two numeric values: `Chi` (Chi-squared statistic) and `Pval` (the P-value).

Author(s)

Alastair Potts <potts.a@gmail.com>

References

Tajima, F. (1993) Simple methods for testing molecular clock hypothesis. *Genetics*, **135**, 599–607. (Equation 4)

Examples

```
require(ape)
data(woodmouse)
rr.test(x = woodmouse[2, ], y = woodmouse[3, ], out = woodmouse[1, ])

# Test all pairs in a sample:
outgroup <- woodmouse[1, ]
n <- nrow(woodmouse)
cc <- combn(2:n, 2)
FUN <- function(x)
    rr.test(woodmouse[x[1], ], woodmouse[x[2], ], outgroup)$Pval
OUT <- apply(cc, 2, FUN)
### two ways to arrange the output:
RES <- matrix(NA, n - 1, n - 1)
RES[row(RES) > col(RES)] <- OUT
RES <- t(RES)
RES[row(RES) > col(RES)] <- OUT
RES <- t(RES)
dimnames(RES) <- list(2:n, 2:n)
RES <- as.dist(RES)
### 2nd method:
class(OUT) <- "dist"
attr(OUT, "Labels") <- as.character(2:15)
attr(OUT, "Size") <- n - 1L
attr(OUT, "Diag") <- attr(OUT, "Upper") <- FALSE
### they are the same:
all(OUT == RES)
```

`site.spectrum` *Site Frequency Spectrum*

Description

`site.spectrum` computes the (un)folded site frequency spectrum of a set of aligned DNA sequences.

Usage

```
site.spectrum(x, folded = TRUE, outgroup = 1)
## S3 method for class 'spectrum'
plot(x, col = "red", main = NULL, ...)
```

Arguments

x	a set of DNA sequences (as an object of class "DNAbin"), or an object of class "spectrum".
folded	a logical specifying whether to compute the folded site frequency spectrum (the default), or the unfolded spectrum if folded = FALSE.
outgroup	a single integer value giving which sequence is ancestral; ignored if folded = TRUE.
col	the colour of the barplot (red by default).
main	a character string for the title of the plot; a generic title is given by default (use main = "" to have no title).
...	further arguments passed to barplot.

Details

Under the infinite sites model of mutation, mutations occur on distinct sites, so every segregating (polymorphic) site defines a partition of the n sequences (see Wakeley, 2009). The *site frequency spectrum* is a series of values where the ith element is the number of segregating sites defining a partition of i and $n-i$ sequences. The *unfolded* version requires to define an ancestral state with an external (outgroup) sequence, so i varies between 1 and $n-1$. If no ancestral state can be defined, the *folded* version is computed, so i varies between 1 and $n/2$ or $(n-1)/2$, for n even or odd, respectively.

If folded = TRUE, sites with more than two states are ignored and a warning is returned giving how many were found.

If folded = FALSE, sites with an ambiguous state at the external sequence are ignored and a warning is returned giving how many were found. Note that it is not checked if some sites have more than two states.

Value

site.spectrum returns an object of class "spectrum" which is a vector of integers (some values may be equal to zero) with the attribute "folded" (a logical value) indicating which version of the spectrum has been computed.

Author(s)

Emmanuel Paradis

References

Wakeley, J. (2009) *Coalescent Theory: An Introduction.* Greenwood Village, CO: Roberts and Company Publishers.

See Also

DNAbin for manipulation of DNA sequences in R, haplotype

Examples

```
require(ape)
data(woodmouse)
(sp <- site.spectrum(woodmouse))
plot(sp)
```

---

summary.loci *Print and Summaries of Loci Objects*

Description

These functions print and summarize table of alleles and loci (objects of class "loci").

Usage

```
## S3 method for class 'loci'
print(x, details = FALSE, ...)
## S3 method for class 'loci'
summary(object, ...)
## S3 method for class 'summary.loci'
print(x, ...)
## S3 method for class 'loci'
x[i, j, drop = TRUE]
## S3 method for class 'summary.loci'
plot(x, loci, what = "both", layout = 1, col = c("blue", "red"), ...)
```

Arguments

x, object	an object of class "loci" or "summary.loci".
details	a logical value: if TRUE the data are printed as a data frame; the default is FALSE.
i, j	indices of the rows and/or columns to select or to drop. They may be numeric, logical, or character (in the same way than for standard R objects).
drop	a logical specifying whether to returned an object of the smallest dimension possible, i.e., may return a vector or a factor if drop = TRUE (this is not the default).
loci	the loci (genes) to be plotted. By default, all loci are plotted.
what	the frequencies to be plotted. Three choices are possible: "alleles", "genotypes", and "both" (the default), or any unambiguous abbreviations.
layout	the number of graphs to be plotted simultaneously.
col	the colours used for the barplots.
...	further arguments to be passed to or from other methods.

Details

Genotypes not observed in the data frame are not counted.

When using the [method, if only one column is extracted or if the returned data frame has no 'loci' column, then the class "loci" is dropped.

An object of class "loci" can be edited in the R data editor with, e.g., fix(x) or x <- edit(x).

summary.loci computes the absolute frequencies (counts); see the examples on how to compute the relative frequencies (proportions).

Value

summary.loci returns a list with the genes as names and each element made a list with two vectors "genotype" and "allele" with the frequencies (numbers) of genotypes and alleles, respectively. The names of these two vectors are the observed genotypes and alleles.

print and plot methods return NULL.

Author(s)

Emmanuel Paradis

See Also

read.loci, getAlleles, edit.loci

Examples

```
require(adegenet)
data(nancycats)
x <- as.loci(nancycats)
s <- summary(x)
plot(s, layout=20, las=2)
layout(1)

## compute the relative frequencies:
rapply(s, function(x) x/sum(x), how = "replace")
```

---

tajima.test *Test of the Neutral Mutation Hypothesis*

Description

This function tests the neutral mutation hypothesis with Tajima's *D*.

Usage

```
tajima.test(x)
```

Arguments

x	a set of DNA sequences (object of class "DNAbin").

Value

A list with three numeric values:

D	Tajima's *D* statistic.
Pval.normal	the p-value assuming that *D* follows a normal distribution with mean zero and variance one.
Pval.beta	the p-value assuming that *D* follows a beta distribution after rescaling on [0, 1] (Tajima, 1989).

Author(s)

Emmanuel Paradis

References

Tajima, F. (1989) Statistical method for testing the neutral mutation hypothesis by DNA polymorphism. *Genetics*, **123**, 595–595.

Examples

```
require(ape)
data(woodmouse)
tajima.test(woodmouse)
```

---

theta.h *Population Parameter THETA using Homozygosity*

Description

This function computes the population parameter THETA using the homozygosity (or mean heterozygosity) from gene frequencies.

Usage

```
theta.h(x, standard.error = FALSE)
```

Arguments

x	a vector or a factor.
standard.error	a logical indicating whether the standard error of the estimated theta should be returned (TRUE), the default being FALSE.

Details

The argument x can be either a factor or a vector. If it is a factor, then it is taken to give the individual alleles in the population. If it is a numeric vector, then its values are taken to be the numbers of each allele in the population. If it is a non-numeric vector, it is a coerced as a factor.

The standard error is computed with an approximation due to Chakraborty and Weiss (1991).

Value

A numeric vector of length one with the estimated theta (the default), or of length two if the standard error is returned (`standard.error = TRUE`).

Author(s)

Emmanuel Paradis

References

Zouros, E. (1979) Mutation rates, population sizes and amounts of electrophoretic variation at enzyme loci in natural populations. *Genetics*, **92**, 623–646.

Chakraborty, R. and Weiss, K. M. (1991) Genetic variation of the mitochondrial DNA genome in American Indians is at mutation-drift equilibrium. *American Journal of Physical Anthropology*, **86**, 497–506.

See Also

heterozygosity, theta.s, theta.k, theta.tree

Examples

```
## similar to what is in ?H:
require(adegenet)
data(nancycats)
## convert the data and compute frequencies:
S <- summary(as.loci(nancycats))
## compute THETA for all loci:
sapply(S, function(x) theta.h(x$allele))
```

---

theta.k *Population Parameter THETA using Expected Number of Alleles*

Description

This function computes the population parameter THETA using the expected number of alleles.

Usage

```
theta.k(x, n = NULL, k = NULL)
```

Arguments

x	a vector or a factor.
n	a numeric giving the sample size.
k	a numeric giving the number of alleles.

Details

This function can be used in two ways: either with a vector giving the individual genotypes from which the sample size and number of alleles are derived (e.g., `theta.k(x)`), or giving directly these two quantities (e.g., `theta.k(n = 50, k = 5)`).

The argument `x` can be either a factor or a vector. If it is a factor, then it is taken to give the individual alleles in the population. If it is a numeric vector, then its values are taken to be the numbers of each allele in the population. If it is a non-numeric vector, it is a coerced as a factor.

Both arguments `n` and `k` must be single numeric values.

Value

A numeric vector of length one with the estimated theta.

Note

For the moment, no standard-error or confidence interval is computed.

Author(s)

Emmanuel Paradis

References

Ewens, W. J. (1972) The sampling theory of selectively neutral alleles. *Theoretical Population Biology*, **3**, 87–112.

See Also

`theta.h`, `theta.s`, `theta.tree`

Examples

```
require(adegenet)
data(nancycats)
## convert the data and compute frequencies:
S <- summary(as.loci(nancycats))
## compute THETA for all loci:
sapply(S, function(x) theta.k(x$allele))
```

`theta.msat` *Population Parameter THETA From Micro-Satellites*

Description

This function estimates the population parameter θ using micro-satellite data with three different estimators.

Usage

```
theta.msat(x)
```

Arguments

`x`	an object of class `"loci"`.

Details

The data must be micro-satellites, so the allele names must be the repeat counts (see the example).

The three estimators are based on (i) the variance of the number of repeats, (ii) the expected homozygosity (both described in Kimmel et al., 1998), and (iii) the mean allele frequencies (Haasl and Payseur, 2010).

Value

a numeric matrix with loci as rows and the three estimates of θ as columns.

Author(s)

Emmanuel Paradis

References

Kimmel, M., Chakraborty, R., King, J. P., Bamshad, M., Watkins, W. S. and Jorde, L. B. (1998) Signatures of population expansion in microsatellite repeat data. *Genetics*, **148**, 1921–1930.

Haasl, R. J. and Payseur, B. A. (2010) The number of alleles at a microsatellite defines the allele frequency spectrum and facilitates fast accurate estimation of θ. *Molecular Biology and Evolution*, **27**, 2702–2715.

See Also

`theta.h`, `theta.tree`

Examples

```
require(adegenet)
data(nancycats)
x <- as.loci(nancycats)
theta.msat(x)
```

theta.s *Population Parameter THETA using Segregating Sites*

Description

This function computes the population parameter THETA using the number of segregating sites s in a sample of n DNA sequences.

Usage

```
theta.s(s, n, variance = FALSE)
```

Arguments

s	a numeric giving the number of segregating sites.
n	a numeric giving the number of sequences.
variance	a logical indicating whether the variance of the estimated THETA should be returned (TRUE), the default being FALSE.

Value

A numeric vector of length one with the estimated theta (the default), or of length two if the standard error is returned (`variance = TRUE`).

Note

The number of segregating sites needs to be computed beforehand, for instance with the function `seg.sites` (see example below).

Author(s)

Emmanuel Paradis

References

Watterson, G. A. (1975) On the number of segragating sites in genetical models without recombination. *Theoretical Population Biology*, **7**, 256–276.

Tajima, F. (1989) Statistical method for testing the neutral mutation hypothesis by DNA polymorphism. *Genetics*, **123**, 585–595.

See Also

theta.h, theta.k, seg.sites, nuc.div, theta.tree

Examples

```
data(woodmouse)
s <- length(seg.sites(woodmouse))
n <- nrow(woodmouse)
theta.s(s, n)
theta.s(s, n, variance = TRUE)
```

---

theta.tree *Population Parameter THETA Using Genealogy*

Description

This function estimates the population parameter θ from a genealogy (coded a as phylogenetic tree) under the coalescent.

Usage

```
theta.tree(phy, theta, fixed = FALSE, analytical = TRUE, log = TRUE)
```

Arguments

phy	an object of class "phylo".
theta	a numeric vector.
fixed	a logical specifying whether to estimate theta (the default), or to return the likelihoods for all values in theta.
analytical	a logical specifying whether to use analytical formulae to estimate theta and its standard-error. If FALSE, a numerical optimisation of the likelihood is performed (this option is ignored if fixed = TRUE).
log	a logical specifying whether to return the likelihoods on a log scale (the default); ignored if fixed = FALSE.

Details

The tree phy is considered as a genealogy, and therefore should be ultrametric. By default, θ is estimated by maximum likelihood and the value given in theta is used as starting value for the minimisation function (if several values are given as a vector the first one is used). If fixed = TRUE, then the [log-]likelihood values are returned corresponding to each value in theta.

The present implementation does a numerical optimisation of the log-likelihood function (with nlminb) with the first partial derivative as gradient. It is possible to solve the latter and have a direct analytical MLE of θ (and its standard-error), but this does not seem to be faster.

Value

If `fixed = FALSE`, a list with two elements:

`theta`	the maximum likelihood estimate of θ;
`logLik`	the log-likelihood at its maximum.

If `fixed = TRUE`, a numeric vector with the [log-]likelihood values.

Author(s)

Emmanuel Paradis

References

Kingman, J. F. C. (1982) The coalescent. *Stochastic Processes and their Applications*, **13**, 235–248.

Kingman, J. F. C. (1982) On the genealogy of large populations. *Journal of Applied Probability*, **19A**, 27–43.

Wakeley, J. (2009) *Coalescent Theory: An Introduction.* Greenwood Village, CO: Roberts and Company Publishers.

See Also

`theta.h`, `theta.s`, `theta.k`

Examples

```
tr <- rcoal(50) # assumes theta = 1
theta.tree(tr, 10)
theta.tree(tr, 10, analytical = FALSE) # uses nlminb()
## profile log-likelihood:
THETA <- seq(0.5, 1.5, 0.01)
logLikelihood <- theta.tree(tr, THETA, fixed = TRUE)
plot(THETA, logLikelihood, type = "l")
```

---

utilities *Utily Functions for pegas*

Description

The first three functions extract information on loci, `expand.genotype` creates a table of all possible genotypes given a set of alleles, `proba.genotype` calculates expected probabilities of genotypes under Hardy–Weinberg equilibrium, and the last two functions test whether a locus is a SNP or whether a gentotype is phased.

Usage

```
getPloidy(x)
getAlleles(x)
getGenotypes(x)
expand.genotype(n, alleles = NULL, ploidy = 2, matrix = FALSE)
proba.genotype(alleles = c("1", "2"), p, ploidy = 2)
is.snp(x)
is.phased(x)
```

Arguments

x	an object of class "loci".
n	an integer giving how many alleles to consider (ignored if alleles is used).
alleles	the allele names as a vector of mode character.
ploidy	an integer giving the ploidy level (either 2 or 4 for the moment).
matrix	a logical specifying whether to return the genotypes in a matrix or as a character vector.
p	a vector of allele probabilities; if missing, equal probabilities are assumed.

Details

expand.genotype and proba.genotype accept any level of ploidy and any number of alleles.

For is.snp, a locus is defined as a SNP if it has two alleles and their labels are made of a single character (e.g., A and T, or 1 and 2, but not A and AT).

Value

getPloidy returns the ploidy level of all loci in an object of class "loci" as a numeric vector.

getAlleles and getGenotypes return the alleles and genotypes, respectively, observed in all loci in an object of class "loci" as a list.

expand.genotype returns a character vector (the default) or a matrix where the rows are the genotypes and the columns are the alleles. The matrix is numeric by default, or character if the argument alleles is given.

proba.genotype returns a numeric vector with names set as the genotypes.

is.snp returns a logical vector specifying whether each locus is a SNP.

is.phased returns a matrix of the same size than the original data specifying whether each genotype is phased or not.

Author(s)

Emmanuel Paradis

Examples

```
require(adegenet)
data(nancycats)
X <- as.loci(nancycats)[, 2:3]
getAlleles(X)
getGenotypes(X)
expand.genotype(2)
expand.genotype(2, LETTERS[1:3])
expand.genotype(3, ploidy = 4)
proba.genotype() # classical HWE with 2 alleles
## an octoploid with a six-allele locus (1287 possible genotypes):
length(p <- proba.genotype(alleles = LETTERS[1:6], ploidy = 8))
max(p) # ~ 0.006
## the cat data:
s <- summary(X)
## allele counts from the first locus:
p <- s[[1]]$allele
## expected probabilities for the 136 possible genotypes...
proba.genotype(names(p), p/sum(p))
## ... to be compared with s[[1]]$genotype
```

---

write.loci *Write Allelic Data Files*

Description

This function writes allelic data into a text file.

Usage

```
write.loci(x, file = "", loci.sep = " ", allele.sep = "/|", ...)
```

Arguments

x	an object of class "loci".
file	a file name specified by either a variable of mode character, or a quoted string. By default, the data are printed on the console.
loci.sep	the character(s) use to separate the loci (columns) in the file (a space by default).
allele.sep	the character(s) used to separate the alleles for each locus in the file (a slash by default).
...	further arguments passed to write.table.

Value

NULL

Author(s)

Emmanuel Paradis

See Also

read.loci, write.table for all its options

Examples

```
require(adegenet)
data(nancycats)
x <- as.loci(nancycats)[1:10, 1:3] # take a small subset
write.loci(x)
## use of '...':
write.loci(x, loci.sep = "\t", quote = FALSE, col.names = FALSE)
```

Index