Package ‘pegas’

March 5, 2015

Version 0.7
Date 2015-03-05
Title Population and Evolutionary Genetics Analysis System
Depends R (>= 2.6.0), ape (>= 2.4), adegenet
Imports graphics, utils
ZipData no
Description Functions for reading, writing, plotting, analysing, and manipulating allelic and haplo-
typic data, and for the analysis of population nucleotide sequences and micro-satellites includ-
ing 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 yes
Repository CRAN
Date/Publication 2015-03-05 22:09:38

R topics documented:

<table>
<thead>
<tr>
<th>R topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>pegas-package</td>
<td>2</td>
</tr>
<tr>
<td>amova</td>
<td>3</td>
</tr>
<tr>
<td>as.loci</td>
<td>5</td>
</tr>
<tr>
<td>bind.loci</td>
<td>6</td>
</tr>
<tr>
<td>diffHaplo</td>
<td>7</td>
</tr>
<tr>
<td>edit.loci</td>
<td>8</td>
</tr>
<tr>
<td>Fst</td>
<td>9</td>
</tr>
<tr>
<td>geod</td>
<td>10</td>
</tr>
<tr>
<td>geoTrans</td>
<td>11</td>
</tr>
</tbody>
</table>
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.html.

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

```r
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 funHx[[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


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

```r
### 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)
```
Description

These functions do conversion among different allelic data classes.

Usage

```r
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

```r
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)
```

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

Usage

```r
## 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".
diffHaplo

Comparison Between Two Haplotypes

Description

This function compares two haplotypes and returns a summary of the differences.

Usage

diffHaplo(h, a = 1, b = 2)

Arguments

h

an object of class "haplotype".

a, b

two integers (or character strings) giving the indices (or labels) of the two haplotypes to be compared.

Details

This function prints the number of transitions and transversions between both sequences, and returns a data frame with three columns giving the positions of the differences and the nucleotides in each sequence at these positions.
Value

A data frame with three columns named `pos` (position of the differences) and the labels of the two haplotypes compared.

Author(s)

Emmanuel Paradis

See Also

`haploNet`, `haplotype`

Examples

data(woodmouse)
h <- haplotype(woodmouse)
diffHaplo(h) # compares the 1st and 2nd haplotypes
diffHaplo(h, 1, 3)
diffHaplo(h, "I", "III") # same than above but using labels

---

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

```r
## 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")`. 
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 feedback are still welcome.

Author(s)

Emmanuel Paradis

References


See Also

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

Examples

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

Description

This function calculates geodesic (or great-circle) distances between pairs of points with their longitudes and latitudes given in (decimal) degrees.

Usage

```r
geod(lon, lat = NULL, R = 6371)
```

Arguments

- `lon` either a vector of numeric values with the longitudes in degrees, or, if `lat = NULL`, a matrix giving the longitudes (first column) and the latitudes (second column).
- `lat` a vector with the latitudes.
- `R` the mean radius of the Earth (see details).

Details

The default value of `R` is the mean radius of the Earth which is slightly smaller than the radius at the equator (6378.1 km).

Value

a numeric symmetric matrix with the distances between pairs of points in kilometres.

Author(s)

Emmanuel Paradis

References

http://en.wikipedia.org/wiki/Great-circle_distance
http://en.wikipedia.org/wiki/Earth
Manipulate Geographic Coordinates

**Description**

This function transforms standard geographical coordinates in degrees, minutes and seconds input as characters (or a factor) into numerical values in degrees.

**Usage**

```r
geoTrans(x, degsym = NULL, minsym = "'", secsym = ")")
```

**Arguments**

- `x`: a vector of character strings storing geographical coordinates; this can be a factor with the levels correctly set.
- `degsym`, `minsym`, `secsym`: a single character giving the symbol used for degrees, minutes and seconds, respectively.

**Details**

This function should be robust to any pattern of spacing around the values and the symbols (see examples).

If the letter S, W, or O is found is the coordinate, the returned value is negative.

Note that longitude and latitude should not be mixed in the same character strings.

The default for `degsym` (NULL) is because the degree symbol (°) is coded differently in different character encodings. By default, the function will use the appropriate character depending on the system and encoding used.

**Value**

a numeric vector with the coordinates in degrees (eventually as decimal values).
Author(s)

Emmanuel Paradis

See Also

geod

Examples

```r
coord <- c("N 43*27.30"", "N43*27.30", "43*27.30\"N", "43* 27' 30" N", "43* 27' 30"\' N", "43*27.30\"", "43*27.5\"")
cat(coord, sep = "\n")
geoTrans(coord)
geoTrans("43 D 27.5\'", degsym = "D")
geoTrans("43* 27' 30\" S")
```

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

**Usage**

```r
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`. 
haploNet

Value

a matrix of counts.

Author(s)

Klaus Schliep and Emmanuel Paradis

See Also

haplotype, haploNet

Examples

```r
## 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(rownames(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)
```

Description

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

Usage

```r
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,
     threshold = c(1, 2), ...)
```
## 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 or a numeric value: if 0 (or FALSE) nothing is drawn on the links; if 1 (or TRUE) the mutations are shown with small dots on the links; if 2 the number of mutations are printed on the links.
- **threshold**: a numeric vector with two values (or 0) giving the lower and upper numbers of mutations for alternative links to be displayed. If threshold = 0, alternative links are not drawn at all.
- **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 Ham-
mong distance) of DNA sequences and pairwise deletion of missing data (see dist.dna) Users may
specify their own distance with the argument d. 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 three additional attributes: "freq", the absolute frequencies of each haplotype, "labels", their
labels, and "alter.links", the alternative links of the network.

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

Note

If two haplotypes are very different, haploNet will likely fail (error during integration due to non-
finite values).

Author(s)

Emmanuel Paradis, Klaus Schliep

References

ation with haplotypes inferred from restriction endonuclease mapping and DNA sequence data. III.
Cladogram estimation. Genetics, 132, 619–635.

See Also

haplotype, haploFreq, replot, diffHaplo, 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)
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).

From pegas 0.7, haplotype extracts haplotypes taking into account base ambiguities.
haplotype.loci

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 `i`th 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


See Also

`thetaNs`

Examples

```r
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))
```
Description

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

Usage

\texttt{hw.test(x, B = 1000)}

Arguments

\texttt{x} \quad \text{an object of class "loci".}

\texttt{b} \quad \text{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 \(\chi^2\)-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 \(\chi^2\)-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

\begin{verbatim}
require(adegenet)
data(nancycats)
x <- as.loci(nancycats)
hw.test(x)
\end{verbatim}
**Description**

These two functions analyse linkage disequilibrium in the case of phased \(LD\) and unphased \(LD_2\) genotypes.

**Usage**

\[
LD(x, \text{locus} = 1:2, \text{details} = \text{TRUE})
LD2(x, \text{locus} = 1:2, \text{details} = \text{TRUE})
\]

**Arguments**

- \(x\): an object of class "loci".
- \(\text{locus}\): a vector of integers giving the two loci to analyse.
- \(\text{details}\): 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 \(\text{details} = \text{FALSE}\), only the \(T_2\) test is returned.

For LD: if \(\text{details} = \text{TRUE}\), 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


See Also

haplotype.loci, is.phased

Examples

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

**Mismatch Distribution**

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

**Usage**

```r
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 (Rogers and Harpending 1992).

**Author(s)**

Emmanuel Paradis and David Winter

**References**


**Examples**

```r
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

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


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 \( \theta \) 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: `rR` 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


See Also

- `read.dna, dist.dna`

Examples

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

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

Usage

```r
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


See Also

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

Examples

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

```r
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-package.ird.fr/pegas/DefinitionDataClassesPegas.pdf

Author(s)

Emmanuel Paradis

See Also

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

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 be compressed (*.gz) or not, but compressed files cannot be read remotely (see examples).

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:

- CHR: the chromosome number (characters);
- POS: the position of the locus (numeric values);
- QUAL: the quality of the inferred locus (integer values);
- FILTER: the filter (characters).

Author(s)

Emmanuel Paradis

References

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

See Also

read.loci, read.gtx, write.loci

Examples

```r
## Not run:
## Chr Y from the 1000 genomes:
b <- "ALL.chrY.phase3_integrated_v1.20130502.genotypes.vcf.gz"
## WARNING: the name of the file above may have changed
url <- paste(a, b, sep = "/")
## file is compressed, so we download first:
download.file(url, "tmp.vcf.gz")
## no need to uncompress to read now that the file is local:
(x <- read.vcf("tmp.vcf.gz", 4000)) # read only 4000 loci

SNP <- is.snp(x)
table(SNP) # how many loci are really SNPs?

op <- par(mfcol = c(4, 1), xpd = TRUE)
lim <- c(2.65e6, 2.95e6)
## distribution of SNP and non-SNP mutations along the Y chr:
plot(attr(x, "POS"), !SNP, "h", col = "red", main = "non-SNP mutations")
rect(lim[1], -0.1, lim[2], 1.1, lwd = 2, lty = 2)
plot(attr(x, "POS"), SNP, "h", col = "blue", main = "SNP mutations")
rect(lim[1], -0.1, lim[2], 1.1, lwd = 2, lty = 2)
par(xpd = FALSE)
## same focusing on a smaller portion of the chromosome:
plot(attr(x, "POS"), !SNP, "h", col = "red", xlim = lim)
plot(attr(x, "POS"), SNP, "h", col = "blue", xlim = lim)
par(op)

## get haplotypes for the first 10 loci:
h <- haplotype(x, 1:10)
```
replot

## Description

This function makes possible to change the layout of a haplotype network interactively or with specified coordinates.

## Usage

```r
replot(xy = NULL, ...)
```

## Arguments

- **xy**
  - an optional list with vectors names `x` and `y` (or `xx` and `yy`) giving the coordinates of the nodes.

- **...**
  - further arguments passed to `plot`.

## Details

This function can be used in two ways. By default (i.e., `replot()`), the user can edit a plotted haplotype network by clicking with the mouse on the graphical window: a message is printed asking to click once close to the node to move and then clicking again where this node should be placed (careful: two separate single clicks). Editing is stopped with a right click.

The second possible use is to specify the new coordinates of the nodes with the argument `xy`, typically, from a previous call to `replot` (see examples).

## Value

A named list with two numeric vectors (`x` and `y`).

## Author(s)

Emmanuel Paradis

## See Also

`haploNet`, `haploFreq`
Examples

```r
## Not run:
data(woodmouse)
net <- haploNet(haplotype(woodmouse))
plot(net)
o <- replot()  # interactive
## click to rearrange the network at will...
## then do a different plot using the same coordinates:
plot(net, bg = "red", labels = FALSE, show.mutation = 2)
replot(o)  # not interactive
## End(Not run)
```

---

**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**

Examples

```r
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

```r
site.spectrum(x, folded = TRUE, outgroup = 1)
```

## S3 method for class 'spectrum'
```r
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

Company Publishers.

See Also

DNAbin for manipulation of DNA sequences in R, haplotype

Examples

require(ape)
data(woodmouse)
(sp <- site.spectrum(woodmouse))
plot(sp)
Description
These functions print and summarize table of alleles and loci (objects of class "loci").

Usage
```r
## 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

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

## compute the relative frequencies:
apply(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.
- $P_{\text{val.normal}}$: the p-value assuming that $D$ follows a normal distribution with mean zero and variance one.
- $P_{\text{val.beta}}$: the p-value assuming that $D$ follows a beta distribution after rescaling on [0, 1] (Tajima, 1989).
Author(s)
Emmanuel Paradis

References

Examples

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

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

Usage

```r
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 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


See Also

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

Examples

```r
## 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

```r
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 coerced as a factor.

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

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


See Also

theta.h, theta.s, theta.tree

Examples

```r
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 $\theta$ 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 $\theta$ as columns.

Author(s)
Emmanuel Paradis

References


See Also
theta.h, theta.tree

Examples
```r
require(adeegenet)
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
```r
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


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)
Description

This function estimates the population parameter $\theta$ from a genealogy (coded as a 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, $\theta$ 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 \texttt{nlminb}) with the first partial derivative as gradient. It is possible to solve the latter and have a direct analytical MLE of $\theta$ (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 $\theta$;
  - **logLik**: the log-likelihood at its maximum.
- If fixed = TRUE, a numeric vector with the [log-]likelihood values.

Author(s)

Emmanuel Paradis
References


See Also

theta.h, theta.s, theta.k

Examples

```r
tr <- rcoal(50) # assumes theta = 1
tree.tree(tr, 10)
tree.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

 Utility 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 genotype is phased.

Usage

```r
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
Description

This function writes allelic data into a text file.

Usage

\[
\text{write.loci}(x, \text{file} = "", \text{loci.sep} = " ", \text{allele.sep} = "/", \ldots)
\]

Arguments

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

Value

\text{NULL}

Author(s)

Emmanuel Paradis

See Also

\text{read.loci, write.table} for all its options

Examples

\[
\begin{align*}
\text{require(adegenet)}\\
\text{data(nancycats)}\\
x & \leftarrow \text{as.loci(nancycats)[1:10, 1:3]} \quad \# \text{take a small subset}\\
\text{write.loci}(x)\\
\text{## use of } \ldots:\
\text{write.loci}(x, \text{loci.sep} = "\t", \text{quote} = \text{FALSE}, \text{col.names} = \text{FALSE})
\end{align*}
\]
Index

*Topic **IO**
   as.loci, 5
   edit.loci, 8
   read.gtx, 27
   read.loci, 28
   read.vcf, 29
   write.loci, 45

*Topic **hplot**
   haploNet, 13
   haplotype, 16
   MMD, 22
   replot, 31
   site.spectrum, 33
   summary.loci, 35

*Topic **htest**
   Fst, 9
   hw.test, 20
   R2.test, 25
   rr.test, 32
   tajima.test, 36

*Topic **manip**
   bind.loci, 6
   diffHaplo, 7
   geod, 10
   geoTrans, 11
   haploFreq, 12
   haplotype, 16
   haplotype.loci, 17
   heterozygosity, 18
   nuc.div, 24
   site.spectrum, 33
   summary.loci, 35
   theta.h, 37
   theta.k, 38
   theta.s, 40
   utilities, 43

*Topic **models**
   amova, 3
   haploNet, 13
   LD, 21
   mst, 23
   theta.msat, 39
   theta.tree, 42

*Topic **package**
   pegas-package, 2

*Topic **univar**
   heterozygosity, 18
   nuc.div, 24
   theta.h, 37
   theta.k, 38
   theta.s, 40
   [.haplotype (haplotype), 16
   [.loci (summary.loci), 35

   adonis, 4
   amova, 3, 4
   as.igraph.haploNet (haploNet), 13
   as.loci, 5
   as.network.haploNet (haploNet), 13

   barplot, 16, 34
   base.freq, 25
   bind.loci, 6
   cbind.loci (bind.loci), 6
   df2genind, 6
   diffHaplo, 7, 15
   dist.dna, 15, 26
   dist.haplotype.loci (haplotype.loci), 17
   DNAbin, 17, 34

   edit.loci, 8, 36
   expand.genotype (utilities), 43

   Fst, 9
   fstat, 10

   GC.content, 25
   genind, 6

46
genind2loci (as.loci), 5

read.vcf, 27, 29, 29

replot, 15, 31

rr.test, 32

rug, 23

seg.sites, 25, 41

site.spectrum, 33

sort.haplotype (haplotype), 16

summary.loci, 9, 29, 35

tajima.test, 36

theta.h, 37, 39–41, 43

theta.k, 38, 41, 43

theta.msat, 39

theta.s, 19, 25, 38, 39, 40, 43

theta.tree, 38–41, 42

utilities, 43

write.loci, 27, 29, 30, 45

write.table, 45