Package 'haplotypes'

July 15, 2023

Type Package

Title Manipulating DNA Sequences and Estimating Unambiguous Haplotype Network with Statistical Parsimony

Version 1.1.3.1

Date 2023-07-12

Author Caner Aktas

Maintainer Caner Aktas <caktas.aca@gmail.com>

Description Provides S4 classes and methods for reading and manipulating aligned DNA sequences, supporting an indel coding methods (only simple indel coding method is available in the current version), showing base substitutions and indels, calculating absolute pairwise distances between DNA sequences, and collapses identical DNA sequences into haplotypes or inferring haplotypes using user provided absolute pairwise character difference matrix. This package also includes S4 classes and methods for estimating genealogical relationships among haplotypes using statistical parsimony and plotting parsimony networks.

License GPL-2

URL https://cran.r-project.org,
https://biolsystematics.wordpress.com/r/

Imports methods, stats, utils, network, sna, ape, phangorn,plotrix

NeedsCompilation no

Repository CRAN

Date/Publication 2023-07-15 05:40:03 UTC

R topics documented:

`haplotypes-package` *Manipulating DNA Sequences and Estimating Unambiguous Haplotype Network with Statistical Parsimony*

Description

This package provides S4 classes and methods for reading and manipulating aligned DNA sequences, supporting an indel coding methods (only simple indel coding method is available in the current version), showing base substitutions and indels, calculating absolute pairwise distances between DNA sequences, and collapses identical DNA sequences into haplotypes or infering haplotypes using user provided absolute pairwise character difference matrix. This package also includes S4 classes and methods for estimating genealogical relationships among haplotypes using statistical parsimony and plotting parsimony networks.

Author(s)

Caner Aktas, <caktas.aca@gmail.com>

Examples

```
## Read example FASTA file.
f<-system.file("example.fas",package="haplotypes")
# invalid character 'N' was replaced with '?' with a warning message
x<-read.fas(file=f)
# an object of class 'Dna'
x

## or load DNA Sequence data set.
data("dna.obj")
x<-dna.obj
## Not run:
x

## End(Not run)

## Compute an absolute pairwise character difference matrix from DNA sequences.
# coding gaps using simple indel coding method
d<- distance(x,indels="sic")
## Not run:
d

## End(Not run)

## Infer haplotypes using the 'Dna' object.
# coding gaps using simple indel coding method
h<-haplotype(x,indels="s")
## Not run:
h

## End(Not run)

## Conduct statistical parsimony analysis with 95% connection limit.
#algortihmic method
## Not run:
p<-parsimnet(x,prob=.95)
```

```
p
# plot network
plot(p)

## End(Not run)

## Plotting pie charts on the statistical parsimony network
## Not run:
data("dna.obj")
x<-dna.obj
h<-haplotypes::haplotype(x)

## Statistical parsimony with 95
p<-parsimnet(x)

#randomly generated populations
pop<-c("pop1","pop2","pop3","pop4","pop5","pop6","pop7","pop8")
set.seed(5)
pops<-sample(pop,nrow(x),replace=TRUE)

# Plotting with default parameters.
pieplot(p,h,1, pops)


## End(Not run)
```

---

append-methods *Combines two* Dna *objects*

Description

Combines two Dna objects.

Usage

```
## S4 method for signature 'Dna'
append(x,values)
```

Arguments

x	an object of class Dna.
values	an object of class Dna.

Value

an object of class Dna.

Methods

`signature(x = "Dna", values= "Dna")` combines two Dna objects.

Examples

```
data("dna.obj")

x<-dna.obj
y<-dna.obj
nrow(x)

## Combining two 'Dna' objects.
z<- append(x,y)
nrow(z)
```

---

as.data.frame-methods *Coerces a* Dna *object to a data.frame*

Description

Coerces an object to a data.frame.

Usage

```
## S4 method for signature 'Dna'
as.data.frame(x)
```

Arguments

x	an object of class Dna.

Value

returns a data frame.

Methods

`signature(x = "Dna")` coerces a Dna object to a data.frame.

Examples

```
data("dna.obj")

x<-dna.obj
x<-as.dna(x[1:4,1:6])

## Coercing a 'Dna' object to a data.frame.
df<-as.data.frame(x)
df

# TRUE
is.data.frame(df)

## Not run:
# gives the same result
df<-as.data.frame(x@sequence)
df

## End(Not run)
```

`as.dna-methods` *Coerces an object to a* `Dna` *object*

Description

Coerces an object that contains DNA sequences to an object of Class `Dna`.

Usage

```
## S4 method for signature 'matrix'
as.dna(x)
## S4 method for signature 'data.frame'
as.dna(x)
## S4 method for signature 'list'
as.dna(x)
## S4 method for signature 'character'
as.dna(x)
## S4 method for signature 'Haplotype'
as.dna(x)
## S4 method for signature 'DNAbin'
as.dna(x)
## S4 method for signature 'phyDat'
as.dna(x)
```

Arguments

x	a matrix, a data.frame, a list, a character, an object of class Haplotype, DNAbin {ape} or phyDat {phangorn} object containing the DNA sequences.

Details

Elements of the list must be vectors. Each element of the list contains a single DNA sequence. If the sequence lengths differ, the longest sequence is taken into account and gaps are introduced to the shorter sequences at the end of the matrix in the slot sequence. Sequence length information is stored in the slot seqlengths.

Valid characters for the slot sequence are "A","C","G","T","a","c","g","t","-","?". During the conversion of the object to the class Dna, integers 0,1,2,3,4,5 or characters "0","1","2","3","4","5" are converted to "?","A","C","G","T","-", respectively. Invalid characters are replaced with "?" with a warning message.

Value

an object of class Dna.

Methods

signature(x = "matrix") coerces matrix to a Dna object.

signature(x = "data.frame") coerces data.frame to a Dna object.

signature(x = "list") coerces list to a Dna object.

signature(x = "character") coerces characters to a Dna object.

signature(x = "Haplotype") coerces a Haplotype object to a Dna object.

signature(x = "DNAbin") coerces a DNAbin object to a Dna object.

signature(x = "phyDat") coerces a phyDat object to a Dna object.

Examples

```
## Coercing a matrix to a 'Dna' object.
# all valid characters
x<-matrix(c("?","A","C","g","t","-","0","1","2","3","4","5"),4,6)
rownames(x)<-c("seq1","seq2","seq3","seq4")
dna.obj<-as.dna(x)
dna.obj
# the sequence matrix
dna.obj@sequence

## Not run:
# includes invalid characters
x<-matrix(c("X","y","*","?","t","-","0","1","2","3","4","5"),4,6)
rownames(x)<-c("seq1","seq2","seq3","seq4")
dna.obj<-as.dna(x)
dna.obj
dna.obj@sequence
```

```
# all valid integers
x<-matrix(c(0,1,2,3,4,5,0,1,2,3,4,5),4,6)
rownames(x)<-c("seq1","seq2","seq3","seq4")
dna.obj<-as.dna(x)
dna.obj
dna.obj@sequence

## Coercing a data.frame to a 'Dna' object.
x<-data.frame(matrix(c("?","A","C","g","t","-","0","1","2","3","4","5"),4,6))
rownames(x)<-c("seq1","seq2","seq3","seq4")
dna.obj<-as.dna(x)
dna.obj
dna.obj@sequence

## Coercing a list to a 'Dna' object.
seq1<-c("?","A","C","g","t","-","0","1")
seq2<-c("?","A","C","g","t","-","0","1","2")
seq3<-c("?","A","C","g","t","-","0","1","2","3")
x<-list(seq1=seq1,seq2=seq2,seq3=seq3)
dna.obj<-as.dna(x)

# sequence lengths differ
dna.obj@seqlengths
dna.obj@sequence

## Coercing a character vector to a Dna object.
seq<-c("?","A","C","g","t","-","0","1")
x<-as.dna(seq)
x


## Coercing a Haplotype object to a Dna object.
data("dna.obj")
x<-dna.obj
h<-haplotype(x)

# DNA Sequences of unique haplotypes
dna.obj<-as.dna(h)
dna.obj

d<-distance(x)

# if 'Haplotype' object does not contain 'DNA' Sequences
h<-haplotype(d)

# returns an error
as.dna(h)

## Coercing a DNAbin object to a Dna object.
require(ape)
data(woodmouse)
x<-as.dna(woodmouse)
```

```
x


## End(Not run)
```

`as.DNAbin-methods` *Coerces an object to a* `DNAbin` *object*

Description

This function coerces `Dna` object to `DNAbin` {ape} object .

Usage

```
## S4 method for signature 'Dna'
as.DNAbin(x, endgaps=TRUE)
```

Arguments

`x`	an object of class `Dna`.
`endgaps`	boolean; gaps at the end of the sequences are included if this is TRUE.

Value

an object of class `DNAbin`.

Methods

`signature(x = "Dna")` coerces a `Dna` object to a `DNAbin` object.

Examples

```
## Coercing a Dna object to a DNAbin object.
data("dna.obj")

x<-dna.obj
dBin<-as.DNAbin(x)
dBin

#gaps at the end removed
dBin<-as.DNAbin(x, endgaps=FALSE)
dBin
```

as.list-methods *Methods for function* as.list *in the Package* **haplotypes**

Description

Coerces an object to a list.

Usage

```
## S4 method for signature 'Dna'
as.list(x)
## S4 method for signature 'Haplotype'
as.list(x)
## S4 method for signature 'Parsimnet'
as.list(x)
```

Arguments

x	an object of class Dna,Haplotype or Parsimnet.

Details

If x is a Dna object, elements of the list are character vectors that contains the DNA sequences of length equal to corresponding value in the slot seqlengths. If x is Haplotype or Parsimnet objects, slots are converted to list elements.

Value

returns a list.

Methods

signature(x = "Dna") coerces an object of class Dna to a list.

signature(x = "Haplotype") coerces an object of class Haplotype to a list.

signature(x = "Parsimnet") coerces an object of class Parsimnet to a list.

Examples

```
data("dna.obj")

## Coercing a 'Dna' object to a list.
x<-dna.obj[1:3,as.matrix=FALSE]
as.list(x)

## Not run:
## Coercing a 'Haplotype' object to a list.
x<-dna.obj
```

```
h<-haplotype(x)
as.list(h)

## Coercing a 'Parsimnet' object to a list.
x<-dna.obj
p<-parsimnet(x)
as.list(p)

## End(Not run)
```

`as.matrix-methods` *Methods for function* `as.matrix` *in the Package* **haplotypes**

Description

Coerces an object to a matrix.

Usage

```
## S4 method for signature 'Dna'
as.matrix(x)
```

Arguments

`x`	an object of class `Dna`.

Value

returns a character matrix.

Methods

`signature(x = "Dna")` coerces an object of class Dna to a matrix.

Examples

```
data("dna.obj")

## Coercing a 'Dna' object to a matrix.
x<-dna.obj[1:4,1:6,as.matrix=FALSE]
x
as.matrix(x)

## Not run:
# gives the same result
dna.obj[1:4,1:6,as.matrix=TRUE]
## End(Not run)
```

as.network-methods *Coerces an object to a* network *object*

Description

This function coerces Parsimnet object to network {network} object .

Usage

```
## S4 method for signature 'Parsimnet'
as.network(x,net=1,...)
```

Arguments

x	an object of class Parsimnet.
net	a numeric vector of length one indicating which network to convert.
...	additional arguments to function network.

Value

an object of class network.

Methods

signature(x = "Parsimnet") coerces a Parsimnet object to a network object.

Examples

```
## Coercing a Parsimnet object to a network object.
data("dna.obj")
x<-dna.obj
p<-parsimnet(x)
n<-as.network(p)

#Fourth network (with only two edges)
p<-parsimnet(x,prob=.99)
n<-as.network(p,net=4)
```

as.networx-methods *Coerces an object to a* networx *object*

Description

This function coerces Parsimnet object to networx {phangorn} object .

Usage

```
## S4 method for signature 'Parsimnet'
as.networx(x,net=1,...)
```

Arguments

x	an object of class Parsimnet.
net	a numeric vector of length one indicating which network to convert.
...	additional arguments to as.splits.

Value

an object of class networx.

Methods

signature(x = "Parsimnet") coerces a Parsimnet object to a networx object.

Examples

```
## Coercing a Parsimnet object to a networx object.
data("dna.obj")
x<-dna.obj
p<-parsimnet(x)
nx<-as.networx(p)
plot(nx, "2D")
```

`as.numeric-methods` *Coerces a* `Dna` *object to a numeric matrix*

Description

Converts a character matrix to a numeric matrix.

Usage

```
## S4 method for signature 'Dna'
as.numeric(x)
```

Arguments

`x`	an object of class `Dna`.

Details

Function `as.numeric()` coerces the character matrix in the slot `sequence` to a numeric matrix. Lower or upper case characters "?","A","C","G","T","-" are converted to integers 0,1,2,3,4,5, respectively.

Value

returns a numeric matrix.

Methods

`signature(x = "Dna")` coerces a Dna object to a numeric matrix.

Examples

```
x<-matrix(c("?","A","C","g","t","-","0","1","2","3","4","5"),4,6)
rownames(x)<-c("seq1","seq2","seq3","seq4")
x<-as.dna(x)
# original character matrix
as.matrix(x)

## Coercing a 'Dna' object to a numeric matrix.
# numeric matrix
as.numeric(x)
```

as.phyDat-methods *Coerces an object to a* phyDat *object*

Description

This function coerces Dna object to phyDat {phangorn} object .

Usage

```
## S4 method for signature 'Dna'
as.phyDat(x, indels="sic",...)
```

Arguments

x	an object of class Dna.
indels	the indel coding method to be used. This must be one of "sic", "5th" or "missing". Any unambiguous substring can be given. See also 'Details'
...	additional arguments to as.phyDat.

Details

Available indel coding methods:

sic: Treating gaps as a missing character and coding them separately following the simple indel coding method.

5th: Treating gaps as a fifth state character.

missing: Treating gaps as a missing character.

Value

an object of class phyDat.

Methods

signature(x = "Dna") coerces a Dna object to a phyDat object.

Examples

```
data("dna.obj")
x<-dna.obj


## Coercing a Dna object to a phyDat object.
# Simple indel coding.
phyd<-as.phyDat(x)
phyd
```

```
# Gaps as 5th state characters.
phyd<-as.phyDat(x,indels="5")
phyd

# Gaps as 5th state characters.
phyd<-as.phyDat(x,indels="m")
phyd
```

---

basecomp-methods *Calculates base composition*

Description

Calculates base composition of Dna object.

Usage

```
## S4 method for signature 'Dna'
basecomp(x)
```

Arguments

x	an object of class Dna.

Value

a matrix with sequence as rows, DNA bases as columns and frequencies as entries.

Methods

`signature(x = "Dna")` calculates base composition of Dna object.

Examples

```
data("dna.obj")
x <-dna.obj

## Calculating base compositions.
basecomp(x)
```

boot.dna-methods *Generates single bootstrap replicate*

Description

Methods for generating a single bootstrap replicate.

Usage

```
## S4 method for signature 'Dna'
boot.dna(x,replacement=TRUE)
```

Arguments

x	an object of class Dna.
replacement	boolean; whether the sampling is done with replacement or without replacement.

Value

an object of class Dna.

Methods

signature(x = "Dna") generates single bootstrap replicate from a Dna object.

Author(s)

Caner Aktas, <caktas.aca@gmail.com>

Examples

```
data("dna.obj")
x<-dna.obj

## Generating a bootstrap replicate.
# with replacement
bxr<-boot.dna(x)
image(bxr)

# without replacement
bx<-boot.dna(x,replacement=FALSE)
image(bx)
```

`distance-methods` *Calculates absolute pairwise character difference matrix using a* `Dna` *object*

Description

Computes and returns an absolute pairwise character difference matrix from DNA sequences.

Usage

```
## S4 method for signature 'Dna'
distance(x,subset=NULL,indels="sic")
```

Arguments

`x`	an object of class `Dna`.
`subset`	a vector of integers in the range [1,nrow(x)], specifying which sequence(s) are used in the distance calculation. Only distance between selected sequence(s) and the rest of the sequences are calculated. If it is NULL, all comparisons are done.
`indels`	the indel coding method to be used. This must be one of "sic", "5th" or "missing". Any unambiguous substring can be given. See also 'Details'

Details

Available indel coding methods:

`sic`**:** Treating gaps as a missing character and coding them separately following the simple indel coding method.

`5th`**:** Treating gaps as a fifth state character.

`missing`**:** Treating gaps as a missing character.

Value

returns an object of class `dist`.

Methods

`signature(x = "Dna")` Computes and returns an absolute pairwise character difference matrix from `Dna` objects.

Author(s)

Caner Aktas, <caktas.aca@gmail.com>

References

Giribet, G. and Wheeler, W.C. (1999) On gaps. *Molecular Phylogenetics and Evolution* **13**, 132-143.

Simmons, M., Ochoterena, H. (2000) Gaps as characters in sequence-based phylogenetic analyses. *Systematic Biology* **49**, 369-381.

See Also

`indelcoder` and `subs`

Examples

```
data("dna.obj")
x<-dna.obj[4:7,13:22,as.matrix=FALSE]

## Simple indel coding.
distance(x,indels="s")

## Gaps as 5th state characters.
distance(x,indels="5")

## Gaps as missing characters.
distance(x,indels="m")

## Not run:
## Using 'subset'.
x<-dna.obj[4:10,13:22,as.matrix=FALSE]
distance(x, NULL)
distance(x, subset=c(1))
distance(x, subset=c(2,4))

## End(Not run)
```

Dna-class *Class* "Dna" *in the Package* **haplotypes**

Description

S4 class to hold DNA sequence data.

Objects from the Class

Objects can be created by calls of the form `new("Dna", sequence, seqlengths, seqnames)`, however reading fasta file using `read.fas` function or coerce matrix, data.frame or list objects to a Dna object using `as.dna` methods is preferable.

Slots

`sequence`**:** Object of class `"matrix"` containing DNA sequence data, rows represent sequences and columns represent sites. See also 'Note'.

`seqlengths`**:** Object of class `"numeric"` containing the length of each DNA sequence.

`seqnames`**:** Object of class `"character"` containing the name of each DNA sequence.

Methods

[`signature(x = "Dna", i = "ANY", j = "ANY")`: extracts part of a DNA sequence as an object of class matrix.

[<- `signature(x = "Dna", i = "ANY", j = "ANY", value = "ANY")`: replaces part of a Dna sequence with an object of class `"matrix"`, `"numeric"` or `"character"`.

append `signature(x = "Dna", value = "ANY")`: combines two Dna objects.

as.data.frame `signature(x = "Dna")`: coerces an object of class Dna to a data.frame.

as.list `signature(x = "Dna")`: coerces an object of class Dna to a list; elements of the list are character vectors that contains the DNA sequences of length equal to corresponding value in the slot `seqlengths`.

as.matrix `signature(x = "Dna")`: coerces an object of class Dna to a matrix.

as.numeric `signature(x = "Dna")`: coerces an object of class Dna to a numeric matrix.

as.DNAbin `signature(x = "Dna")`: coerces an object of class Dna to a `DNAbin` object.

as.phyDat `signature(x = "Dna")`: coerces an object of class Dna to a `phyDat` object.

basecomp `signature(x = "Dna")`: calculates base composition of Dna object.

boot.dna `signature(x = "Dna")`: generates single bootstrap replicate.

distance `signature(x = "Dna")`: computes and returns an absolute pairwise character difference matrix from DNA sequences.

haplotype `signature(x = "Dna")`: infers haplotypes from DNA sequences.

image `signature(x = "Dna")`: displays DNA sequences

indelcoder `signature(x = "Dna")`: supports simple indel coding method.

length `signature(x = "Dna")`: returns the longest sequence length.

append `signature(x = "Dna", value = "ANY")`: combines two Dna objects.

names `signature(x = "Dna")`: gets the names of an object Dna.

names<- `signature(x = "Dna")`: sets the names of an object Dna.

ncol `signature(x = "Dna")`: returns the longest sequence length.

nrow `signature(x = "Dna")`: returns the total sequence number.

pairnei `signature(x = "Dna")`: calculates pairwise Nei's average number of differences between populations.

pairPhiST `signature(x = "Dna")`: calculates pairwise PhiST between populations.

parsimnet `signature(x = "Dna")`: estimates genealogies using statistical parsimony.

polymorp `signature(x = "Dna")`: displays information about DNA polymorphisms of two sequences; indels and base substitutions, respectively.

range `signature(object = "Dna")`: returns a vector containing the minimum and maximum length of DNA sequences.

remove.gaps `signature(object = "Dna")`: removes alignment gaps.

rownames `signature(object = "Dna")`: retrieve the row names of a DNA sequence matrix.

rownames<- `signature(object = "Dna")`: set the row names of a DNA sequence matrix.

show `signature(object = "Dna")`: displays Dna object briefly.

subs `signature(x = "Dna")`: displays information about base substitutions.

tolower `signature(x = "Dna")`: Translate characters in DNA sequence matrix from upper to lower case.

toupper `signature(x = "Dna")`: Translate characters in DNA sequence matrix from lower to upper case.

unique `signature(x = "Dna")`: returns a list with duplicate DNA sequences removed.

Note

Valid characters for the slot `sequence` are "A","C","G","T","a","c","g","t","-","?". Numeric entries (integers) between 0-5 will be converted to "?","A","C","G","T","-", respectively. Invalid characters will be replaced with "?" with a warning message.

Author(s)

Caner Aktas, <caktas.aca@gmail.com>

---

dna.obj *Example DNA sequence data*

Description

An example object of the class `Dna`.

Usage

```
data(dna.obj)
```

Format

`dna.obj` contains a `Dna` object.

Examples

```
data(dna.obj)
dna.obj
```

grouping-methods *Groups haplotypes according to the grouping variable (populations, species, etc.)*

Description

Function for creating a matrix with haplotypes as rows, grouping factor (populations, species, etc.) as columns and abundance as entries.

Usage

```
## S4 method for signature 'Haplotype'
grouping(x,factors)
```

Arguments

x	an object of class Haplotype.
factors	a vector or factor giving the grouping variable (populations, species, etc.), with one element per individual.

Value

a list with two components:

hapmat: a matrix with haplotypes as rows, levels of the grouping factor (populations, species, etc.) as columns and abundance as entries.

hapvec: a vector giving the haplotype identities of individuals.

Methods

signature(x = "Haplotype")

Author(s)

Caner Aktas, <caktas.aca@gmail.com>

See Also

haplotype

Examples

```
data("dna.obj")
x<-dna.obj[1:6,,as.matrix=FALSE]
# inferring haplotypes from DNA sequences
h<-haplotype(x)

## Grouping haplotypes.
```

```
# character vector 'populations' is a grouping factor.
populations<-c("pop1","pop1","pop2","pop3","pop3","pop3")

# length of the argument 'factor' is equal to the number of sequences
g<-grouping(h,factors=populations)
g
```

Haplotype-class *Class* "Haplotype" *in the Package* **haplotypes**

Description

S4 class to store haplotype information.

Objects from the Class

Objects can be created by calls of the form `new("Haplotype", haplist, hapind, uniquehapind, sequence, d, freq, nhap)`, however use function `haplotype` instead.

Slots

`haplist`**:** Object of class "list", containing the names of individuals that share the same haplotype.

`hapind`**:** Object of class "list", containing the index of individuals that share the same haplotype.

`uniquehapind`**:** Object of class "numeric", containing the index of the first occurrence of unique haplotypes.

`sequence`**:** Object of class "matrix" if present, giving the DNA sequence matrix of unique haplotypes.

`d`**:** Object of class "matrix", giving the absolute pairwise character difference matrix of unique haplotypes.

`freq`**:** Object of class "numeric", giving the haplotype frequencies.

`nhap`**:** Object of class "numeric", giving the total number of haplotypes.

Methods

as.dna `signature(x = "Haplotype")`: if Haplotype object contains dna sequences, coerces an object of class Haplotype to an object of class Dna, else returns an error message.

as.list `signature(x = "Haplotype")`: assigns slots of an object Haplotype to list elements.

grouping `signature(x = "Haplotype")`: creates a matrix with haplotypes as rows, grouping factor (populations, species, etc.) as columns and abundance as entries.

hapreord `signature(x = "Haplotype")`: reorders haplotypes according to the ordering factor.

length `signature(x = "Haplotype")`: returns the number of haplotypes.

pieplot `signature(x = "Parsimnet", y = "Haplotype")`: plot pie charts on statistical parsimony network.

pielegend `signature(x = "Parsimnet", y = "Haplotype")`: add legends to pie charts produced using `pieplot`.

show `signature(object = "Haplotype")`: displays the object briefly.

Author(s)

Caner Aktas, <caktas.aca@gmail.com>

`haplotype-methods` *Methods for function* `haplotype` *in the package* **haplotypes**

Description

Collapses identical DNA sequences into haplotypes or infering haplotypes using user provided absolute pairwise character difference matrix.

Usage

```
## S4 method for signature 'Dna'
haplotype(x,indels="sic")
## S4 method for signature 'dist'
haplotype(x)
## S4 method for signature 'matrix'
haplotype(x)
```

Arguments

`x`	an object of class `Dna`, `dist`, or `matrix`.
`indels`	the indel coding method to be used. This must be one of "sic", "5th" or "missing". Any unambiguous substring can be given. See `distance` for details.

Value

`haplotype` returns an object of class `Haplotype`, `as.list-methods` can be used to coerce the object to a list.

Methods

`signature(x = "Dna")` Inferring haplotypes from DNA sequences.

`signature(x = "dist")` Inferring haplotypes using an absolute pairwise character difference matrix (dist object).

`signature(x = "matrix")` Inferring haplotypes using an absolute pairwise character difference matrix.

Author(s)

Caner Aktas, <caktas.aca@gmail.com>

Examples

```
data("dna.obj")
x<-dna.obj[1:6,,as.matrix=FALSE]

##Inferring haplotypes using 'Dna' object.
# coding gaps using simple indel coding method
h<-haplotype(x,indels="sic")
h

# giving DNA sequences of haplotypes
as.dna(h)

## Not run:

## Slots of an object Haplotype
h@haplist  #haplotype list (names)
h@hapind  #haplotype list (index)
h@uniquehapind  #getting index of the first occurrence of haplotypes
h@sequence  #DNA sequences of haplotypes
h@d  #distance matrix of haplotypes
h@freq  #haplotype frequencies
h@nhap  #total number of haplotypes

## End(Not run)

## Inferring haplotypes using dist object.
d<-distance(x)
h<-haplotype(d)
h
## Not run:
# returns an error message
as.dna(h)

## End(Not run)

## Inferring haplotypes using distance matrix.
d<-as.matrix(distance(x))
h<-haplotype(d)
h
## Not run:
# returns an error message
as.dna(h)

## End(Not run)
```

---

hapreord-methods *Reorders haplotypes according to the ordering factor*

Description

Reorders haplotypes according to the ordering factor.

Usage

```
## S4 method for signature 'Haplotype'
hapreord(x,order=c(1:x@nhap))
```

Arguments

x	an object of class Haplotype.
order	a vector giving the order of haplotypes, with one element per haplotype.

Value

returns an object of class Haplotype.

Methods

signature(x = "Haplotype") Reorders haplotypes.

Author(s)

Caner Aktas, <caktas.aca@gmail.com>

See Also

haplotype

Examples

```
data("dna.obj")
x<-dna.obj[1:6,,as.matrix=FALSE]
# inferring haplotypes from DNA sequences
h<-haplotype(x)

## Reordering haplotypes.

# length of the argument 'order' is equal to the number of haplotypes
rh<-hapreord(h,order=c(4,3,1,2))
rh
```

homopoly-methods *Provides the list of homoplastic indels and substitutions*

Description

This function returs the list of homoplastic indels and substitutions.

Usage

```
## S4 method for signature 'Dna'
homopoly(x,indels="sic",...)
```

Arguments

x	an object of class Dna.
indels	the indel coding method to be used. This must be one of "sic", "5th" or "missing". Any unambiguous substring can be given. See distance for details.
...	additional arguments to parsimnet.

Value

a list with following components:

indels	a character vector of homoplastic indels sitewise Consistency Index, names of the character vector gives the site of homoplastic indel.
subs	a character vector of homoplastic substitutions sitewise Consistency Index, names of the character vector gives the site of substitution.

Methods

signature(x = "Dna")

Author(s)

Caner Aktas, <caktas.aca@gmail.com>

Examples

```
data("dna.obj")

### Method for signature 'Dna'.
x<-dna.obj
homopoly(x)
```

image-methods *Display DNA Sequence*

Description

Display an image of DNA sequences .

Usage

```
## S4 method for signature 'Dna'
image(x,all=FALSE,fifth=TRUE,
col=c("#BFBFBF","#0B99FD","#FD0B0B","#11A808","#F5FD0B","#F8F8FF"),
chars=TRUE,cex=1,show.names=TRUE,show.sites=TRUE,xlab="",ylab="",...)
```

Arguments

x	an object of class Dna.
all	boolean; should entire sequence be displayed or only the polymorphic sites?
fifth	boolean; if all==FALSE, should gaps be displayed?
col	an integer or character vector for the colors. By default it is blue for "A", red for "C", green for "G", yellow for "T", white for "-", and grey for "?".
chars	boolean; should characters be displayed on image?
cex	a numeric vector of expansion factor for characters.
show.names	boolean; should sequence names be displayed on the left side.
show.sites	boolean; should site labels be displayed on the bottom side.
xlab	a title for the x axis.
ylab	a title for the y axis.
...	additional arguments to image.default.

Methods

signature(x = "Dna") Display an image of Dna objects

Author(s)

Caner Aktas, <caktas.aca@gmail.com>.

See Also

image.default

Examples

```
data("dna.obj")
x<-dna.obj

## Display only polymorphic sites without gaps
image(x,all=FALSE,fifth=FALSE,show.names=TRUE,cex=0.6)

## Display only polymorphic sites with gaps
image(x,all=FALSE,fifth=TRUE,show.names=TRUE,cex=0.6)

## Not run:
## Display entire sequences
image(x,all=FALSE,show.names=TRUE,cex=0.6)

## End(Not run)
```

---

indelcoder-methods *Codes gaps*

Description

Function for coding gaps separately. Only simple indel coding method is available in the current version.

Usage

```
## S4 method for signature 'Dna'
indelcoder(x)
```

Arguments

x	an object of class Dna.

Value

a list with two components:

`indels`**:** a matrix giving the indel positions (beginnings and ends) and lengths.

`codematrix`**:** a binary matrix giving the indel codings. Missing values are denoted by -1.

Methods

`signature(x = "Dna")` Function for coding gaps separately.

Author(s)

Caner Aktas, <caktas.aca@gmail.com>

References

Simmons, M., Ochoterena, H. (2000) Gaps as characters in sequence-based phylogenetic analyses. *Systematic Biology* **49**, 369-381.

See Also

distance

Examples

```
data("dna.obj")
x<-dna.obj

## Simple indel coding.
indelcoder(x)
```

`length-methods` *Methods for function* `length` *in the package* **haplotypes**

Description

Methods for function `length`.

Usage

```
## S4 method for signature 'Dna'
length(x)
## S4 method for signature 'Haplotype'
length(x)
## S4 method for signature 'Parsimnet'
length(x)
```

Arguments

x	an object of class Dna, Haplotype or Parsimnet.

Value

returns a non-negative integer vector.

Methods

`signature(x = "Dna")` returns the longest sequence length.

`signature(x = "Haplotype")` returns the number of haplotypes.

`signature(x = "Parsimnet")` returns the length of network(s).

See Also

ncol-methods

Examples

```
data("dna.obj")
x<-dna.obj

## Longest sequence length
length(x)


## Total number of haplotypes
h<-haplotype(x)
length(h)

## Length of network(s)
p<-parsimnet(x,prob=.95)
# length of the network
length(p)

p<-parsimnet(x,prob=.99)
# length of the networks
length(p)
```

---

names-methods *Function to get or set names of a* Dna *object or* Parsimnet *object*

Description

Function to get or set sequence names of Dna object or names of network in Parsimnet object.

Usage

```
## S4 method for signature 'Dna'
names(x)
## S4 method for signature 'Parsimnet'
names(x)
## S4 replacement method for signature 'Dna'
names(x)<-value
## S4 replacement method for signature 'Parsimnet'
names(x)<-value
```

Arguments

x	an object of class Dna or Parsimnet.
value	a character vector of the same length as number of sequence or networks.

Methods

`signature(x = "Dna")` Function to get or set names of an object of Dna.

`signature(x = "Parsimnet")` Function to get or set names of networks in Parsimnet object.

Examples

```
data("dna.obj")

x<-dna.obj
x<-as.dna(x[1:4,1:6])

## Getting sequence names.
names(x)

## Setting sequence names.
names(x)<-c("u","v","z","y")
names(x)



x<-dna.obj
p<-parsimnet(x,prob=.99)

##Getting network names in parsimnet object
names(p)
## Setting network names names.
names(p)<-c("a","b","c","d","f","g")
names(p)
```

---

ncol-methods *Returns the length of the longest DNA sequence*

Description

`ncol` returns the number of columns present in a matrix.

Usage

```
## S4 method for signature 'Dna'
ncol(x)
```

Arguments

`x`	an object of class `Dna`.

Value

an integer of length one.

Methods

`signature(x = "Dna")` `ncol` returns the number of columns present in the sequence matrix (length of the longest DNA sequence).

See Also

`length-methods`

Examples

```
data("dna.obj")
x <-dna.obj

## Giving the length of the longest sequence.
ncol(x)
# gives the same result
length(x)
```

`nrow-methods` *Returns the number of DNA sequences*

Description

`nrow` returns the number of rows present in a matrix.

Usage

```
## S4 method for signature 'Dna'
nrow(x)
```

Arguments

`x`	an object of class `Dna`.

Value

an integer of length one.

Methods

`signature(x = "Dna")` `nrow` returns the number of rows present in the sequence matrix (number of sequences).

Examples

```
data("dna.obj")
x <-dna.obj

## Giving the number of sequences.
nrow(x)
```

`pairnei-methods` *Provides the average number of pairwise Nei's (D) differences between populations*

Description

Function provides pairwise Nei's raw number of nucleotide differences between populations.

Usage

```
## S4 method for signature 'Dna'
pairnei(x,populations,indels="sic",nperm=99, subset=NULL,showprogbar=FALSE)
## S4 method for signature 'dist'
pairnei(x,populations,nperm=99, subset=NULL,showprogbar=FALSE)
## S4 method for signature 'matrix'
pairnei(x,populations,nperm=99, subset=NULL,showprogbar=FALSE)
```

Arguments

`x`	an object of class `Dna`, `"dist"` or `"matrix"`.
`populations`	a vector giving the populations, with one element per individual.
`indels`	the indel coding method to be used. This must be one of "sic", "5th" or "missing". Any unambiguous substring can be given. See `distance` for details.
`nperm`	the number of permutations. Set this to 0 to skip the permutation procedure.
`subset`	a vector of integers in the range [1, `length(unique(populations))`], only distances between selected population(s) and the rest of the populations are calculated. If it is NULL, all comparisons are done.
`showprogbar`	boolean; whether the progress bar is displayed or not displayed.

Details

The null distribution of pairwise Nei's differences under the hypothesis of no difference between the populations is obtained by permuting individuals between populations.

Value

a list with following components:

neidist	a matrix giving the average number of pairwise Nei's (D) differences between populations (below diagonal elements) and average number of pairwise differences within populations (diagonal elements).
p	a matrix giving the p-values, or NULL if permutation test is not performed.

Methods

`signature(x = "Dna")`

`signature(x = "dist")`

`signature(x = "matrix")`

Author(s)

Caner Aktas, <caktas.aca@gmail.com>

References

Nei, M. and Li, W. H. (1979) Mathematical model for studying genetic variation in terms of restriction endonucleases. *Proceedings of the National Academy of Sciences of the United States of America* **76**, 5269-5273.

Examples

```
data("dna.obj")


### Method for signature 'Dna'.
x<-dna.obj
x<-dna.obj[c(1,20,21,26,27,28,30,3,4,7,13,14,15,16,23,24,25),,as.matrix=FALSE]
populations<-c("pop1","pop1","pop1","pop1","pop1","pop1","pop1","pop2",
"pop2","pop2","pop2","pop2","pop3","pop4","pop3","pop4","pop4")

##skip permutation testing
pn<-pairnei(x, populations, nperm=0)
pn

#Between populations
as.dist(pn$neidist)

#Within populations
diag(pn$neidist)

##Gaps as missing characters.
pn <-pairnei(x, populations, indels="m", nperm=0)
pn

##using subset, third population against others
```

```
pn<-pairnei(x, populations, nperm=0,subset=c(3))
pn

## Not run:
## 999 permutations.
pn<-pairnei(x, populations, nperm=999, showprogbar=TRUE)
pn

## random populations
x<-dna.obj
populations<-sample(1:4,nrow(x),replace=TRUE)
pn<-pairnei(x, populations, nperm=999, showprogbar=TRUE)
pn

## populations based on clusters
x<-dna.obj
d<-distance(x)
hc<-hclust(d,method="ward.D")
populations<-cutree(hc,4)
pn<-pairnei(x, populations, nperm=999, showprogbar=TRUE)
pn

## End(Not run)

### Method for signature 'dist'.
x<-dna.obj
x<-dna.obj[c(1,20,21,26,27,28,30,3,4,7,13,14,15,16,23,24,25),,as.matrix=FALSE]
populations<-c("pop1","pop1","pop1","pop1","pop1","pop1","pop1","pop2",
"pop2","pop2","pop2","pop2","pop3","pop4","pop3","pop4","pop4")
d<-distance(x)
pn<-pairnei(d, populations,nperm=0)
pn

### Method for signature 'matrix'.
x<-dna.obj
x<-dna.obj[c(1,20,21,26,27,28,30,3,4,7,13,14,15,16,23,24,25),,as.matrix=FALSE]
populations<-c("pop1","pop1","pop1","pop1","pop1","pop1","pop1","pop2",
"pop2","pop2","pop2","pop2","pop3","pop4","pop3","pop4","pop4")
d<-as.matrix(distance(x))
pn<-pairnei(d, populations,nperm=0)
pn
```

`pairPhiST-methods` *Provides the pairwise PhiST between populations*

Description

This function calculates pairwise PhiST between populations using the AMOVA framework.

Usage

```
## S4 method for signature 'Dna'
pairPhiST(x,populations,indels="sic",nperm=99, negatives=FALSE, subset =NULL,
showprogbar=TRUE)
## S4 method for signature 'dist'
pairPhiST(x,populations,nperm=99, negatives=FALSE, subset=NULL,showprogbar=TRUE)
## S4 method for signature 'matrix'
pairPhiST(x,populations,nperm=99,negatives=FALSE, subset=NULL,showprogbar=TRUE)
```

Arguments

x	an object of class Dna, "dist" or "matrix".
populations	a vector giving the populations, with one element per individual.
indels	the indel coding method to be used. This must be one of "sic", "5th" or "missing". Any unambiguous substring can be given. See distance for details.
nperm	the number of permutations. Set this to 0 to skip the permutation procedure.
negatives	boolean; if it is FALSE all negative PhiST values are replaced with zero.
subset	a vector of integers in the range [1, length(unique(populations))], only distances between selected population(s) and the rest of the populations are calculated. If it is NULL, all comparisons are done.
showprogbar	boolean; whether the progress bar is displayed or not displayed.

Details

The null distribution of pairwise PhiST under the hypothesis of no difference between the populations is obtained by permuting individuals between populations.

Value

a list with following components:

PhiST	a matrix giving the PhiST values between populations.
p	a matrix giving the p-values, or NULL if permutation test is not performed.

Methods

signature(x = "Dna")

signature(x = "dist")

signature(x = "matrix")

Note

An internal code Statphi is taken from package **ade4** version 1.7-8 without any modification, author Sandrine Pavoine. Function amova from package **pegas** is used internally to estimate variance components, author Emmanuel Paradis.

Author(s)

Caner Aktas, <caktas.aca@gmail.com>

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.

Examples

```
data("dna.obj")

### Method for signature 'Dna'.
x<-dna.obj
x<-dna.obj[c(1,20,21,26,27,28,30,3,4,7,13,14,15,16,23,24,25),,as.matrix=FALSE]
populations<-c("pop1","pop1","pop1","pop1","pop1","pop1","pop1","pop2",
"pop2","pop2","pop2","pop2","pop3","pop4","pop3","pop4","pop4")

##skip permutation testing
pst<-pairPhiST(x, populations, nperm=0)
pst

##allow negative PhiST values
pst<-pairPhiST(x, populations, nperm=0, negatives=TRUE)
pst

##Gaps as missing characters.
pst<-pairPhiST(x, populations, indels="m", nperm=0, negatives=TRUE)
pst

##using subset, second population against others
pst <-pairPhiST(x, populations, nperm=0,subset=c(2))
pst


## Not run:
## 999 permutations.
pst<-pairPhiST(x, populations, nperm=999,showprogbar=TRUE)
pst

## random populations
x<-dna.obj
populations<-sample(1:4,nrow(x),replace=TRUE)
pst<-pairPhiST(x, populations, nperm=999,showprogbar=TRUE)
pst

## populations based on clusters
x<-dna.obj
d<-distance(x)
hc<-hclust(d,method="ward.D")
populations<-cutree(hc,4)
```

```
pst<-pairPhiST(x, populations, nperm=999,showprogbar=TRUE)
pst

## End(Not run)

### Method for signature 'dist'.
x<-dna.obj
x<-dna.obj[c(1,20,21,26,27,28,30,3,4,7,13,14,15,16,23,24,25),,as.matrix=FALSE]
populations<-c("pop1","pop1","pop1","pop1","pop1","pop1","pop1","pop2",
"pop2","pop2","pop2","pop2","pop3","pop4","pop3","pop4","pop4")
d<-distance(x)
pst<-pairPhiST(d, populations, nperm=0)
pst

### Method for signature 'matrix'.
x<-dna.obj
x<-dna.obj[c(1,20,21,26,27,28,30,3,4,7,13,14,15,16,23,24,25),,as.matrix=FALSE]
populations<-c("pop1","pop1","pop1","pop1","pop1","pop1","pop1","pop2",
"pop2","pop2","pop2","pop2","pop3","pop4","pop3","pop4","pop4")
d<-as.matrix(distance(x))
pst<-pairPhiST(d, populations, nperm=0)
pst
```

`Parsimnet-class` *Class* `"Parsimnet"` *in the Package* **haplotypes**

Description

S4 class to store statistical parsimony networks and additional information.

Objects from the Class

Objects can be created by calls of the form `new("Parsimnet", d, tempProbs, conlimit, prob, nhap, rowindex)`, however use function `parsimnet` instead.

Slots

`d`: Object of class `"list"` containing the geodesic distance matrix of haplotypes and intermediates for each network.

`tempProbs`: Object of class `"numeric"` giving the probabilities of parsimony for mutational steps beyond the connection limit.

`conlimit`: Object of class `"numeric"` giving the number of maximum connection steps at connection limit.

`prob`: Object of class `"numeric"` giving the user defined connection limit.

`nhap`: Object of class `"numeric"` giving the number of haplotypes in each network.

`rowindex`: Object of class `"list"` containing vectors giving the index of haplotypes in each network.

Methods

as.list `signature(x = "Parsimnet")`: assigns slots of an object Parsimnet to list elements.

as.network `signature(x = "Parsimnet")`: coerces `Parsimnet` object to `network` {network} object

as.networx `signature(x = "Parsimnet")`: coerces `Parsimnet` object to `networx` {phangorn} object

length `signature(x = "Parsimnet")`: returns the length of network(s).

names `signature(x = "Parsimnet")`: gets names of networks in `Parsimnet` object

names<- `signature(x = "Parsimnet")`: sets names of networks in `Parsimnet` object

plot `signature(x = "Parsimnet")`: plots statistical parsimony networks.

pieplot `signature(x = "Parsimnet", y = "Haplotype")`: plots pie charts on statistical parsimony networks

pielegend `signature(x = "Parsimnet", y = "Haplotype")`: add legends to pie charts produced using `pieplot`.

rownames `signature(x = "Parsimnet")`: gets names of vertices in networks.

rownames<- `signature(x = "Parsimnet")`: gets names of vertices in networks

show `signature(object = "Parsimnet")`: displays the object briefly.

Author(s)

Caner Aktas, <caktas.aca@gmail.com>

`parsimnet-methods` *Estimates gene genealogies using statistical parsimony*

Description

Function for estimating gene genealogies from DNA sequences or user provided absolute pairwise character difference matrix using statistical parsimony.

Usage

```
## S4 method for signature 'Dna'
parsimnet(x,indels="sic",prob=.95)
## S4 method for signature 'dist'
parsimnet(x,seqlength,prob=.95)
## S4 method for signature 'matrix'
parsimnet(x,seqlength,prob=.95)
```

Arguments

x	an object of class Dna, dist, or matrix.
indels	the indel coding method to be used. This must be one of "sic", "5th" or "missing". Any unambiguous substring can be given. See distance for details.
seqlength	an integer of length one giving the sequence length information (number of characters).
prob	a numeric vector of length one in the range [0.01, 0.99] giving the probability of parsimony as defined in Templeton et al. (1992). In order to set maximum connection steps to Inf (to connect all the haplotypes in a single network), set the probability to NULL.

Details

The network estimation methods implemented in parsimnet function finds one of the most parsimonious network (or sub-networks if connection between haplotypes exceeds the parsimony limit). This is an implemetation of the TCS method proposed in Templeton et al. (1992) and Clement et al. (2002).parsimnet function generates an unambiguous haplotype network without loops. If more than one best networks found (results in ambiguous connections), only a network with the lowest average all-pairs distance is returned. Loops may occur only if they are present in initial haplotype distance matrix.

Value

S4 methods for signature 'Dna', 'matrix' or 'dist' returns an object of class Parsimnet.

Methods

signature(x = "Dna") estimating gene genealogies from DNA sequences.

signature(x = "dist") estimating gene genealogies from distance matrix (dist object).

signature(x = "matrix") estimating gene genealogies from distance matrix.

Note

Duplicate names in the final distance matrices in slot d are renamed without warning. An internal function .TempletonProb is taken from package pegas version 0.6 without any modification, authors Emmanuel Paradis, Klaus Schliep.

Author(s)

Caner Aktas, <caktas.aca@gmail.com>.

References

Clement, M., Q. Snell, P. Walker, D. Posada, and K. A. Crandall (2002) TCS: Estimating Gene Genealogies *in First IEEE International Workshop on High Performance Computational Biology (HiCOMB)*

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

See Also

network, plot-methods and pieplot-methods

Examples

```
## Not run:
data("dna.obj")
x<-dna.obj

### Method for signature 'Dna'.
## statistical parsimony with 95
p<-parsimnet(x)
p
plot(p)

## statistical parsimony with 99
p<-parsimnet(x,prob=.99)
p
# plot the first network
plot(p,net=1)

## statistical parsimony with 99
#indels are coded as missing
p<-parsimnet(x,indels="m",prob=.99)
p
plot(p)


# statistical parsimony without connection limit.
p<-parsimnet(x,prob=NULL)
p
plot(p)


# plot the first network
plot(p,net=1)

### Method for signature 'dist'.
d<-distance(x)
seqlength<-length(x)
## statistical parsimony with 95
p<-parsimnet(d,seqlength)
p
plot(p)

### Method for signature 'matrix'.
```

```
d<-as.matrix(distance(x))
seqlength<-length(x)
## statistical parsimony with 95
p<-parsimnet(d,seqlength)
p
plot(p)


## End(Not run)
```

---

pielegend-methods *Add Legends to Plots*

Description

This function can be used to add legends to pie charts produced using pieplot.

Usage

```
## S4 method for signature 'Parsimnet,Haplotype'
pielegend(p,h,net=1,factors,...)
```

Arguments

p	a Parsimnet object.
h	a Haplotype object.
net	a numeric vector of length one indicating which network to plot.
factors	a vector or factor giving the grouping variable (populations, species, etc.), with one element per individual.
...	arguments to be passed to legend and others. See 'Details'

Details

This method calls legend {graphics}, some default parameters changed:

col an integer or character vector for the edge colors. By default, it is rainbow.

fill an integer or character vector for the filling colors. By default, it is rainbow

legend a numeric or character vector to appear in the legend. By default, it is the levels of the grouping factor for haplotypes.

x position of the legend. Default is set to "topright".

Value

See ' legend {graphics}'

Methods

signature(p = "Parsimnet", h = "Haplotype")

Author(s)

Caner Aktas, <caktas.aca@gmail.com>.

See Also

plot,Parsimnet-method,floating.pie, plot.default, plot.network.default and legend.

Examples

```
data("dna.obj")
x<-dna.obj
h<-haplotypes::haplotype(x)

### Statistical parsimony with 95% connection limit
p<-parsimnet(x)

#randomly generated populations
pop<-c("pop1","pop2","pop3","pop4","pop5","pop6","pop7","pop8")
set.seed(5)
pops<-sample(pop,nrow(x),replace=TRUE)

## Plotting with default parameters.
pieplot(p,h,1, pops)

## Add legend with default parameters.
pielegend(p,h,1,pops)

## Change colors for the populations.
#8 colors for 8 populations
cols<-colors()[c(30,369,552,558,538,642,142,91)]
pieplot(p,h,1, pops,col=cols)
pielegend(p,h,1,pops,col= cols)
```

---

pieplot-methods *Plots pie charts on statistical parsimony network*

Description

Plotting pie charts on the statistical parsimony network.

Usage

```
## S4 method for signature 'Parsimnet,Haplotype'
pieplot(x,y,net=1,factors, coord = NULL,inter.labels=FALSE,interactive=FALSE,rex=1,...)
```

Arguments

x	a Parsimnet object.
y	a Haplotype object.
net	a numeric vector of length one indicating which network to plot.
factors	a vector or factor giving the grouping variable (populations, species, etc.), with one element per individual.
coord	a matrix that contains user specified coordinates of the vertices, or NULL to generate vertex layouts with network.layout function from package **network**.
inter.labels	boolean; should vertex labels of intermediate haplotypes be displayed?
interactive	boolean; should vertices be interactively adjusted?
rex	expansion factor for the pie radius.
...	arguments to be passed to floating.pie and others. See 'Details'

Details

This method calls floating.pie {plotrix}, network.vertex {network}, and plot.default, lines, and text {graphics}. This method also uses some internal structures of plot.network.default from package **network**. The following additional arguments can be passed to these functions:

mode the vertex placement algorithm. Default is set to "fruchtermanreingold".

pad amount to pad the plotting range; useful if labels are being clipped. Default is set to 1.

displaylabels boolean; should vertex labels be displayed?

label a vector of vertex labels. By default, the rownames of the distance matrix (rownames(p@d[[net]])) are used. If inter.labels==FALSE only haplotype labels are displayed.

label.cex character expansion factor for labels. Default is set to 0.75.

label.col an integer or character vector for the label colors. By default, it is 1 (black).

label.pos position at which labels should be placed relative to vertices. 0 and 6 results in labels which are placed away from the center of the plotting region; 1, 2, 3, and 4 result in labels being placed below, to the left of, above, and to the right of vertices, respectively; and label.pos 5 or greater than 6 results in labels which are plotted with no offset (i.e., at the vertex positions). Default is set to 0.

label.pad amount to pad the labels. This setting is available only if the labels are plotted with offset relative to vertex positions. Default is set to 1.

vertex.cex a numeric vector of expansion factor for intermediate vertices (only). By default it is (0.5)*min(radius). Use 'radius' to specify size of pie charts.

col the colors of the pie sectors (i.e., colors for populations), by default rainbow.

vertex.col an integer or character vector for the intermediate vertex colors. By default, it is 1 (black).

edge.col an integer or character vector for the edge colors. By default, it is 1 (black).

edge.lwd a numeric vector, edges line width. By default, it is 1.

edge.lty a numeric vector of length one, specifies the line type for the edges. By default it is 1.

edges the number of lines forming a pie circle, By default, it is 200.

radius a numeric vector of length p@nhap[net] for the radius of drawn pie charts. Useful for specifying the radius independent of the haplotype frequencies. Default is (0.8*(haplotype frequencies)*rex)/max(haplotype frequencies).

vertex.sides number of polygon sides for vertices. Default is set to 50.

xlab x axis label.

xlab y axis label.

Value

A two-column matrix containing the vertex positions as x,y coordinates.

Methods

signature(x = "Parsimnet", y = "Haplotype")

Note

Some internal structures of plot.network.default is taken from package **network** with modifications, author Carter T. Butts.

Author(s)

Caner Aktas, <caktas.aca@gmail.com>.

See Also

plot,Parsimnet-method,floating.pie, plot.default and plot.network.default

Examples

```
data("dna.obj")
x<-dna.obj
h<-haplotypes::haplotype(x)

### Statistical parsimony with 95% connection limit
p<-parsimnet(x)

#randomly generated populations
pop<-c("pop1","pop2","pop3","pop4","pop5","pop6","pop7","pop8")
set.seed(5)
pops<-sample(pop,nrow(x),replace=TRUE)

## Plotting with default parameters.
pieplot(p,h,1, pops)

## Change colors for the populations.
#8 colors for 8 populations
cols<-colors()[c(30,369,552,558,538,642,142,91)]
pieplot(p,h,1, pops,col=cols)

## Expanding pie charts and intermediate vertices.
```

```
pieplot(p,h,1, pops,rex=2)

## Adjusting intermediate vertex sizes.
pieplot(p,h,1, pops, vertex.cex=rep(0.2, nrow(p@d[[1]])-p@nhap))

## Expanding pie charts and intermediate vertices, adjusting intermediate vertex sizes.
pieplot(p,h,1, pops,rex=2, vertex.cex=rep(0.1, nrow(p@d[[1]])-p@nhap))

## Adjusting radius of pie charts.
pieplot(p,h,1, pops,radius=rep(1, p@nhap))

## Not run:
## Interactively adjusting vertex positions.
pieplot(p,h,1, pops, interactive=TRUE)

## End(Not run)

### Multiple networks with 99% connection limit.
p<-parsimnet(x,prob=.99)

## Plotting first network with default parameters.
pieplot(p,h,1, pops)

## Change colors for the populations.
#8 colors for 8 populations
cols<-colors()[c(30,369,552,558,538,642,142,91)]
pieplot(p,h,1, pops,col=cols)
```

`plot-methods` *Methods for function* `plot` *in the package* **haplotypes**

Description

Plots statistical parsimony networks.

Usage

```
## S4 method for signature 'Parsimnet'
plot(x,y,net=1,inter.labels=FALSE,...)
```

Arguments

x	an object of class Parsimnet.
y	not used (needed for compatibility with generic plot function).
net	a numeric vector of length one indicating which network to plot.
inter.labels	boolean; should vertex labels of intermediate haplotypes to be displayed?
...	additional arguments to plot.default, plot.network.default and arguments to be passed to plot method for Parsimnet objects. See 'Details'

Details

These methods call plot.network.default from package **network**. Some default parameters are changed:

label a vector of vertex labels. By default the row names of the distance matrices in slot d are used. If inter.labels==FALSE only haplotype labels are displayed.

usearrows boolean; should arrows (rather than line segments) be used to indicate edges? Default is set to FALSE.

mode the vertex placement algorithm. Default is set to "kamadakawai".

pad amount to pad the plotting range; useful if labels are being clipped. Default is set to 1.

label.cex character expansion factor for label text. Default is set to 0.75.

vertex.cex a numeric vector of expansion factor for vertices. By default it is 0.8 for haplotypes and 0.5 for intermediates.

vertex.col an integer or character vector for the vertex colors. By default it is 2 (red) for haplotypes and 4 (blue) for intermediates.

Value

A two-column matrix containing the vertex positions as x,y coordinates.

Methods

signature(x = "Parsimnet",y = "missing") Plots Parsimnet objects.

Author(s)

Caner Aktas, <caktas.aca@gmail.com>.

See Also

parsimnet, plot.default and plot.network.default

Examples

```
## Not run:

data("dna.obj")
x<-dna.obj


### Method for signature 'Parsimnet'.

## Statistical parsimony with 95
p<-parsimnet(x)
p
## Plotting with default parameters.
plot(p)

## Displaying vertex labels of intermediate haplotypes.
plot(p, inter.labels=TRUE)

## Interactively adjusting vertex positions.
plot(p, interactive=TRUE)

## Interactively adjusting and saving vertex positions.
p<-parsimnet(x)
#saving vertex positions as x,y coordinates.
coo<-plot(p,interactive=TRUE)
#reuse saved coordinates.
plot(p,coord=coo)

## Adjusting vertex sizes.
plot(p, vertex.cex=c(rep(3,nrow(p@d[[1]]))))
# different sizes for haplotypes and intermediates
plot(p, vertex.cex=c(rep(3,p@nhap),rep(1,c(nrow(p@d[[1]])-p@nhap))))

## Adjusting vertex colors
# different color for haplotypes and intermediates
plot(p, vertex.col=c(rep("magenta",p@nhap),rep("deepskyblue3",c(nrow(p@d[[1]])-p@nhap))))

## Statistical parsimony with 98
p<-parsimnet(x,prob=.99)
p
#plot the first network
plot(p,net=1)
#plot the second network
plot(p,net=2)
#plot the third network. It is a single vertex.
plot(p,net=3)
```

```
## End(Not run)
```

`polymorp-methods` *Displays polymorphic sites (base substitutions and indels) between two sequences*

Description

This function displays the polymorphic sites (base substitutions and indels) between the two sequences.

Usage

```
## S4 method for signature 'Dna'
polymorp(x,pair,indels="sic")
```

Arguments

`x`	an object of class `Dna`.
`pair`	a vector of integers in the range [1,nrow(x)] of length two, specifying sequence pair.
`indels`	the indel coding method to be used. This must be one of "sic", "5th" or "missing". Any unambiguous substring can be given. See `distance` for details.

Value

a list with two components:

`indels`**:** a list of matrices of the indel regions if `indels=="sic"`. The component names of the list gives the position of the indels.

`subst`**:** a list of matrices of the base substitutions. If `indels=="5th"`, each gap is treated as a base substitution. The component names of the list gives the position of the base substitutions.

Methods

`signature(x = "Dna")` Showing base substitutions and indels between the two sequences.

Author(s)

Caner Aktas, `<caktas.aca@gmail.com>`

See Also

`indelcoder` and `subs`

Examples

```
data("dna.obj")
x<-dna.obj

## Showing base substitutions and indels between seq1 and seq6.

# gaps are coded following the simple indel coding method
polymorp(x,c(1,6),indels="s")

# gaps are coded as a fifth state character
polymorp(x,c(1,6),indels="5")

# gaps are treated as missing character
polymorp(x,c(1,6),indels="m")
```

---

range-methods *Returns the minimum and maximum lengths of the DNA sequences*

Description

range returns the lengths of shortest and longest DNA sequences.

Usage

```
## S4 method for signature 'Dna'
range(x)
```

Arguments

x	an object of class Dna.

Value

an integer of length two.

Methods

signature(x = "Dna") range

See Also

length-methods

Examples

```
data("dna.obj")
x <-dna.obj

## shortest and longest DNA sequence lengths
range(x)
```

`read.fas` *Read sequences from a file in FASTA format*

Description

Read DNA sequences from a file in FASTA Format.

Usage

```
read.fas(file)
```

Arguments

`file`	the name of the file, which the sequence in the FASTA format is to be read from. If it does not contain an *absolute* path, the file name is *relative* to the current working directory, `getwd()`.

Value

`read.fas` returns an object of class `Dna`.

Note

By default, valid characters are "A","C","G","T","a","c","g","t","-","?" for the class `Dna`. Numeric entries (integers) between 0-5 will be converted to "?","A","C","G","T","-", respectively. Invalid characters will be replaced with "?" with a warning message.

Author(s)

Caner Aktas, <caktas.aca@gmail.com>

See Also

`Dna`

Examples

```
##Reading example file.
f<-system.file("example.fas",package="haplotypes")

# invalid character 'N' was replaced with '?' with a warning message
x<-read.fas(file=f)

# an object of class 'Dna'
x
```

`remove.gaps-methods` *Removing gaps from* `Dna` *object*

Description

Removing gaps("-") from Dna object

Usage

```
## S4 method for signature 'Dna'
remove.gaps(x,entire.col=FALSE)
```

Arguments

`x`	an object of class Dna.
`entire.col`	boolean; entire columns with gaps are removed if this is TRUE. See also 'Details'.

Details

If entire.col==TRUE, alignment is preserved. If it is FALSE, end gaps are introduced to sequence matrix.

Value

an object of class Dna.

Methods

`signature(x = "Dna")`

Author(s)

Caner Aktas, <caktas.aca@gmail.com>

Examples

```
data("dna.obj")

## original data
x<-dna.obj
range(x)
x@seqlengths

## Only gaps '-' are removed from sequences.
x<-remove.gaps(dna.obj, entire.col=FALSE)
range(x)
x@seqlengths

## entire columns with gaps are removed.
x<-remove.gaps(dna.obj, entire.col=TRUE)
range(x)
x@seqlengths
```

`rownames-methods` *Retrieve or set the row names*

Description

Function to get or set row names of a sequence matrix in a `Dna` object or distance matrix (or matrices) in a `Parsimnet` object.

Usage

```
## S4 method for signature 'Dna'
rownames(x)
## S4 method for signature 'Parsimnet'
rownames(x)
## S4 replacement method for signature 'Dna'
rownames(x)<-value
## S4 replacement method for signature 'Parsimnet'
rownames(x)<-value
```

Arguments

`x`	an object of class `Dna` or `Parsimnet`.
`value`	a character vector of the same length as number of sequences or a list of the same length as number of networks including vertex names for each network. See 'Examples'

Methods

signature(x = "Dna")

signature(x = "Parsimnet")

Examples

```
data("dna.obj")
x<-dna.obj

### Method for signature 'Dna'.

## Getting sequence names.
rownames(x)

## Setting sequence names.
rownames(x)<-c(1:nrow(x))
rownames(x)

### Method for signature 'Parsimnet'.
x<-dna.obj

##single network
p<-parsimnet(x)

##Getting vertex names
rownames(p)

## Setting vertex names.
rownames(p)<-list(c(1:nrow(p@d[[1]])))
rownames(p)
plot(p)

## Multiple networks with 99% connection limit.
p<-parsimnet(x,prob=.99)

## Getting vertex names
rownames(p)

## Setting vertex names.
rownames(p)<-list(1:9, 10, 11,12:13,14,15:16)
rownames(p)
```

---

show-methods *Methods for function* show *in the package* **haplotypes**

Description

Show objects of classes Dna, Haplotype, and Parsimnet

Methods

signature(object = "Dna") displays Dna object briefly: The total number of DNA sequences, names of the first six sequences (if nrow(x)>=6), length of the shortest and longest sequences and the names of the slots.

signature(object = "Haplotype") displays Haplotype object briefly: The list of individuals that share the same haplotypes, the total number of haplotypes and the names of the slots.

signature(object = "Parsimnet") displays Parsimnet object briefly: The total number of networks, the maximum connection steps at chosen probability, the total number of haplotypes in each network,the total number of intermediates in each network, total network lengths (scores) of each network and the names of the slots.

---

subs-methods *Displays base substitutions*

Description

This function displays all base substitutions. If fifth=="TRUE", each gap is treated as a fifth state character.

Usage

```
## S4 method for signature 'Dna'
subs(x,fifth=FALSE)
```

Arguments

x	an object of class Dna.
fifth	boolean; should gaps be treated as a fifth state character?

Value

a list with three components:

subsmat: a sequence matrix showing substitutions.

subs: a list of matrices of the substitutions.

subsmnum: total number of substitutions.

Methods

signature(x = "Dna")

Author(s)

Caner Aktas, <caktas.aca@gmail.com>.

Examples

```
data("dna.obj")
x<-dna.obj

## Base substitutions.
subs(x)

## Gaps are treated as a fifth state character.
subs(x,fifth=TRUE)
```

---

tolower-methods, toupper-methods

Convert sequence characters from upper to lower case or vice versa

---

Description

Convert sequence characters in a Dna object from upper to lower case or vice versa.

Usage

```
## S4 method for signature 'Dna'
tolower(x)
## S4 method for signature 'Dna'
toupper(x)
```

Arguments

x	an object of class Dna.

Value

an object of class Dna.

Methods

signature(x = "Dna")

Examples

```
## Coercing a list to a 'Dna' object.
seq1<-c("?","A","C","g","t","-","0","1")
seq2<-c("?","A","C","g","t","-","0","1","2")
seq3<-c("?","A","C","g","t","-","0","1","2","3")
x<-list(seq1=seq1,seq2=seq2,seq3=seq3)
dna.obj<-as.dna(x)

#characters in Dna object
table(as.matrix(dna.obj))

##all lower case
lowc<-tolower(dna.obj)
#characters
table(as.matrix(lowc))

##all upper case
upc<-toupper(dna.obj)
#characters
table(as.matrix(upc))
```

---

unique-methods *Extract Unique Sequences*

Description

unique returns a list with duplicate sequences removed.

Usage

```
## S4 method for signature 'Dna'
unique(x,gaps=FALSE)
```

Arguments

x	an object of class Dna.
gaps	boolean; gaps are removed if this is FALSE.

Details

This function behaves somehow similar to haplotype, however indels and missing characters are not taken into account.

Methods

signature(x = "Dna")

Examples

```
data("dna.obj")
x<-dna.obj[1:6,,as.matrix=FALSE]

##gaps removed.
unique(x)

##gaps not removed.
unique(x,gaps=TRUE)

##unique vs. haplotype
#unique returns 5 unique sequences.
unique(x)
length(unique(x))

#haplotype returns 4 unique haplotypes with simple indel coding.
h<-haplotype(x)
as.list(as.dna(h))
length(h)

#haplotype returns 3 unique haplotypes with gaps as missing.
h<-haplotype(x, indels="m")
as.list(as.dna(h))
length(h)
```

[-methods *Extract or replace parts of an object of class* Dna

---

Description

Operators acting on sequence matrix to extract or replace parts.

Usage

```
## S4 method for signature 'Dna'
x[i, j,..., drop = FALSE]
## S4 replacement method for signature 'Dna'
x[i, j]<- value
```

Arguments

x	an object of class Dna
i, j	elements to extract or replace.
...	Additional arguments. In this context, ... is used primarily for as.matrix, which is a boolean. If as.matrix is TRUE (default), the function returns a matrix. If drop is also TRUE, and the subset has either a single row or column, the function

will return a vector instead. If as.matrix is FALSE, the function returns an object of class Dna. The as.matrix argument should be specified as as.matrix=TRUE or as.matrix=FALSE within the call. If it is not specified, the function defaults to TRUE.

`drop` boolean; if TRUE and a single row or column is selected, the function returns a vector instead of a matrix. This is only applicable when as.matrix is TRUE.

`value` a character vector or a character matrix.

Details

The S4 method dispatch mechanism matches arguments based on those specified in the signature of the corresponding generic function. However, for some generics that include '...' in their signature, additional arguments can be incorporated in specific methods. Notably, the '[' function does not follow this pattern and restricts the arguments to those defined in its signature. In this context, the 'as.matrix' argument is not in the signature of the generic '[', so it is included within '...'. Then, within the body of the function, we check whether 'as.matrix' has been provided in the actual arguments when the function is called. If 'as.matrix' is not specified, the function defaults to 'TRUE', preserving the behavior of previous versions of the method.

Value

returns an object of class matrix, vector or Dna.

Methods

`signature(x = "Dna", i = "ANY", j = "ANY", drop = "ANY")`

Author(s)

Caner Aktas, <caktas.aca@gmail.com>

See Also

Dna

Examples

```
data("dna.obj")
x<-dna.obj

## Extract parts.
# a matrix object
x[1:2,1:3]

# a Dna object, as.dna(x[1:2,1:3]) gives the same result
x[1:2,1:3,as.matrix=FALSE]

# a vector object
x[1,1:4,drop=TRUE]
```

```
## Replace parts.
#"G" "C"
x[1,1:2]
x[1,1:2]<-c("A","T")
x[1,1:2]
```

Index