# Package 'haplotypes'

April 26, 2015

**Title** Haplotype Inference and Statistical Analysis of Genetic Variation

**Version** 1.0

**Date** 2015-04-24

**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 inferring haplotypes from DNA sequences or user provided absolute character difference matrix. This package also includes S4 classes and methods for estimating genealogical relationships among haplotypes using statistical parsimony.

**License** GPL-2

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

**Depends** R (>= 3.1.1), methods, network

**Suggests** sna

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2015-04-26 09:49:28

## R topics documented:

---

`haplotypes-package` *Haplotype inference and statistical analysis of genetic variation*

---

## Description

This package provides S4 classes and methods for reading and manipulating aligned DNA sequences, supporting indel coding methods, showing polymorphic sites (base substitutions and indels), calculating absolute pairwise distances between DNA sequences, and inferring haplotypes from DNA sequences or user provided absolute character difference matrix. This package also includes S4 classes and methods for estimating genealogical relationships among haplotypes using statistical parsimony. It requires R-packages: **network** to plot statistical 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.
p<-parsimnet(x,prob=.95)

## Not run:
p
# plot network
plot(p)

## 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 'Haplotype'
as.dna(x)
```

## Arguments

| | |
|---|---|
| `x` | a matrix, a data.frame, a list or an object of class `Haplotype` 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 = "Haplotype")` coerces a `Haplotype` 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 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)

## End(Not run)
```

---

`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.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)
```

`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"`.

**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.

**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.

**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 Nei's average number of differences 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.

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

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

## 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,factor)
```

### Arguments

| | |
|---|---|
| x | an object of class Haplotype. |
| factor | 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, grouping factor (populations, species, etc.) as columns and abundance as entries.

hapvec: a vector giving the haplotype number for each individual.

### 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,factor=populations)

# length of the returned component 'hapvec' is equal to the number of sequences
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,` 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.

**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

Inferring haplotypes from DNA sequences or 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
```

---

# 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*

---

## Description

Function to get or set names of an object.

## Usage

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

## Arguments

| | |
|---|---|
| `x` | an object of class `Dna`. |
| `value` | a character vector of the same length as nrow(x). |

## Methods

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

## 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)
```

---

`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")
## S4 method for signature 'dist'
pairnei(x,populations)
## S4 method for signature 'matrix'
pairnei(x,populations)
```

## 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. |

## Value

a list with following components:

| | |
|---|---|
| Between | a matrix giving the average number of pairwise Nei's (D) differences between the populations. |
| Within | a numeric vector giving the average number of pairwise Nei's (D) differences within the populations. |
| Uniquepopulations | a character vector giving the names of the populations. |

## 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")
x<-dna.obj[1:6,,as.matrix=FALSE]
populations<-c("pop1","pop1","pop2","pop3","pop3","pop3")

## Method for signature 'Dna'.
pairnei(x, populations)

## Method for signature 'dist'.
d<-distance(x)
pairnei(d, populations)

## Method for signature 'matrix'.
d<-as.matrix(distance(x))
pairnei(d, populations)
```

---

# 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), how`

## Slots

**d:** Object of class "list" containing the absolute pairwise character difference 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.

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

**plot** `signature(x = "Parsimnet", y = "ANY")`: plots statistical parsimony networks.

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

## Author(s)

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

## References

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.

---

`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 between 0 and 1 giving the probability of parsimony as defined in Templeton et al. (1992). |

## Details

The network estimation method implemented in parsimnet function finds one of the most parsimonious network within the parsimony limit. If algorithm finds more than one best networks, only the minimum spanning network is returned.

## Value

`parsimnet` 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

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

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.

Paradis, E. (2010) pegas: an R package for population genetics with an integrated-modular approach. *Bioinformatics*, **26**, 419-420.

## See Also

`network` and `plot-methods`

## Examples

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

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

# statistical parsimony with %99 connection limit
p<-parsimnet(x,prob=.99)
```

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

# statistical parsimony with %95 connection limit, indels are coded as missing
p<-parsimnet(x,indels="m")
p
plot(p)


## Method for signature 'dist'.
d<-distance(x)
seqlength<-length(x)
# statistical parsimony with %95 connection limit for 113 character
p<-parsimnet(d,seqlength)
p
plot(p)


## Method for signature 'matrix'.
d<-as.matrix(distance(x))
seqlength<-length(x)
# statistical parsimony with %95 connection limit for 113 character
p<-parsimnet(d,seqlength)
p
plot(p)
```

---

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

---

## Description

Plots statistical parsimony networks.

## Usage

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

## Arguments

| | |
|---|---|
| `x` | an object of class `Parsimnet`. |
| `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` and `plot.network.default`. |

## Details

This method calls `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 = "ANY")` Plots Parsimnet objects.

## Author(s)

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

## See Also

`plot.default` and `plot.network.default`

## Examples

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

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

## Plotting with default parameters.
plot(p)

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

## Not run:
## Interactively adjusting vertex positions.
```

```
plot(p, interactive=TRUE)

## End(Not run)

## 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))))

## Statistical parsimony with %98 connection limit.
p<-parsimnet(x,prob=.98)
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)
```

---

`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")
```

---

# 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
```

---

## 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 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)
```

---

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

## Arguments

| | |
|---|---|
| x | an object of class Dna |
| i, j | elements to extract or replace. |
| as.matrix | boolean; If TRUE, function returns a matrix, else, function returns an object of class Dna. |
| value | a character vector or a character matrix. |

## Value

returns an object of class matrix 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]

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

# Index