

# Package: fingerprint (via r-universe)

June 4, 2026

**Version** 3.5.10

**Date** 2026-04-03

**Title** Functions to Operate on Binary Fingerprint Data

**Author** Rajarshi Guha [aut], Zach Charlop-Powers [cre]

**Maintainer** Zach Charlop-Powers <zach.charlop.powers@gmail.com>

**BugReports** <https://github.com/CDK-R/cdkr/issues>

**Description** Functions to manipulate binary fingerprints of arbitrary length. A fingerprint is represented by an object of S4 class 'fingerprint' which is internally represented a vector of integers, such that each element represents the position in the fingerprint that is set to 1. The bitwise logical functions in R are overridden so that they can be used directly with 'fingerprint' objects. A number of distance metrics are also available (many contributed by Michael Fadock). Fingerprints can be converted to Euclidean vectors (i.e., points on the unit hypersphere) and can also be folded using OR. Arbitrary fingerprint formats can be handled via line handlers. Currently handlers are provided for CDK, MOE and BCI fingerprint data.

**License** GPL | file LICENSE

**Depends** methods

**LazyLoad** yes

**Suggests** RUnit, testthat (>= 3.0.0)

**Config/testthat/edition** 3

**NeedsCompilation** yes

**Repository** <https://zachcp.r-universe.dev>

**Date/Publication** 2026-04-04 06:15:38 UTC

**RemoteUrl** <https://github.com/cran/fingerprint>

**RemoteRef** HEAD

**RemoteSha** d81e315b6a1edfb4ff7a829edf4d3fb1a55714b3

## Contents

as.character . . . . .	2
balance . . . . .	3
bit.importance . . . . .	4
bit.spectrum . . . . .	5
c . . . . .	6
cdk.lf, moe.lf, bci.lf . . . . .	6
count-methods . . . . .	7
distance-methods . . . . .	8
euc.vector . . . . .	11
feature-class . . . . .	12
feature-methods . . . . .	13
featvec-class . . . . .	13
fingerprint-class . . . . .	14
fold . . . . .	16
fp.factor.matrix . . . . .	17
fp.read, fp.read.to.matrix . . . . .	18
fp.sim.matrix . . . . .	19
fp.to.matrix . . . . .	20
fplogical . . . . .	21
length . . . . .	21
random.fingerprint . . . . .	22
shannon . . . . .	23
show . . . . .	23

<b>Index</b>	<b>25</b>
--------------	-----------

---

as.character	<i>Generates a String Representation of a Fingerprint</i>
--------------	---

---

### Description

The function returns a string of 1's and 0's or a character vector of features depending on the nature of the fingerprint supplied.

### Usage

```
## S4 method for signature 'fingerprint'
as.character(x)
## S4 method for signature 'featvec'
as.character(x)
## S4 method for signature 'feature'
as.character(x)
```

### Arguments

x	An object of class fingerprint, featvec or feature
---	--

**Value**

A string of 1's and 0's or else a character vector of features (with their counts)

**Author(s)**

Rajarshi Guha <rajarshi.guha@gmail.com>

**Examples**

```
# make a fingerprint vector
fp <- new("fingerprint", nbit=32, bits=sample(1:32, 20))

# print out the string representation
as.character(fp)
```

---

balance

*Generate a Balanced Code Fingerprint*

---

**Description**

It has been noted that the bit density in a fingerprint can affect its ability to retrieve similar compounds from a database primarily due to complexity effects. One approach to alleviating these effects is to generate fingerprints that have a bit density of 50 balanced code approach described by Nisius and Bajorath to convert an ordinary binary fingerprint (whose bit density is not 50 50 (resulting in a fingerprint twice the size of the original).

**Usage**

```
balance(fplist)
```

**Arguments**

fplist            A single fingerprint or a list of fingerprints

**Value**

A single fingerprint objects or list of fingerprint objects that are "balanced", in that they have a bit density of 50 fingerprints.

**Author(s)**

Rajarshi Guha <rajarshi.guha@gmail.com>

**References**

Nisius, B.; Bajorath, J.; *ChemMedChem*, **2010**, *5*, 859-868.

**See Also**

[bit.spectrum](#), [bit.importance](#)

---

bit.importance	<i>Evaluate the Discriminatory Power of Individual Bits in a Binary Fingerprint</i>
----------------	---

---

### Description

This method evaluates the Kullback-Leibler (KL) divergence to rank the individual bits in a binary fingerprint in their ability to discriminate between database and active compounds. This method is implemented based on Nisius and Bajorath and includes an m-estimate correction.

### Usage

```
bit.importance(actives, background)
```

### Arguments

actives	A list of fingerprints for the actives
background	A list of fingerprints representing the background collection

### Value

A numeric vector of length equal to the size of the fingerprints. Each element of the vector is the KL divergence for the corresponding bit. If a bit position is never set to 1 in any of the compounds from the actives and the background, then the KL divergence for that position is undefined and NA is returned.

### Author(s)

Rajarshi Guha <rajarshi.guha@gmail.com>

### References

Nisius, B.; Bajorath, J.; *ChemMedChem*, **2010**, *5*, 859-868.

### See Also

[bit.spectrum](#)

---

`bit.spectrum`*Generate a Bit Spectrum from a List of Fingerprints*

---

### Description

The idea of comparing datasets using fingerprints was described in Guha & Schurer (2008). The idea is that one can summarize the dataset by counting the frequency of occurrence of each bit position. The frequency is normalized by the number of fingerprints considered. Thus a collection of N fingerprints can be converted to a single vector of numbers highlighting the most frequent bits with respect to a given dataset. A plot of this vector looks like a traditional spectrum and hence the name.

The bit spectra for two datasets (assuming that the same types of fingerprints have been used) allows one to compare the similarity of the datasets, without having to do a full pairwise similarity calculation. The difference between the structural features of the datasets can be quantified by evaluating the distance between the two bit spectra.

### Usage

```
bit.spectrum(fplist)
```

### Arguments

`fplist` A list structure with each element being an object of class fingerprint. These will can be constructed by hand or read from disk via [fp.read](#). All fingerprints in the list should be of the same length.

### Value

A numeric vector of length equal to the size of the fingerprints.

### Author(s)

Rajarshi Guha <rajarshi.guha@gmail.com>

### References

Guha, R.; Schurer, S.; *J. Comp. Aid. Molec. Des.*, **2008**, 22, 367-384.

### See Also

[distance](#), [fp.read](#)

---

`c`*Combine Multiple Features to Give a List of Features*

---

**Description**

Combine multiple feature objects to give a list of feature objects

**Usage**

```
## S4 method for signature 'feature'  
c(x, ..., recursive = FALSE)
```

**Arguments**

<code>x</code>	An object of class <code>feature</code>
<code>...</code>	One or more feature objects
<code>recursive</code>	Ignored

**Author(s)**

Rajarshi Guha <rajarshi.guha@gmail.com>

---

`cdk.lf, moe.lf, bci.lf`*Functions to parse lines from fingerprint files*

---

**Description**

These functions take a single line and parse it to produce a vector of integers which represents the position of the 'on' bits in a fingerprint. This allows the user to use `read.fp` with arbitrary fingerprint files. A new file format can be handled by defining a new line parser function. Currently the first three functions process fingerprint files obtained from the CDK (<https://cdk.sourceforge.net>), MOE (<https://www.chemcomp.com>), BCI (<http://www.digitalchemistry.co.uk/>) and the FPS format (<https://code.google.com/archive/p/chem-fingerprints/wikis/FPS.wiki>). The last function can be used for any fingerprint that generates hashed features (such as ECFPs or other circular fingerprints). For these cases, it is assumed that features are unsigned integers, so string features are not handled.

Note that when the `fps.lf` function is specified, items such as the number of bits or the header flag do not need to be specified, as the format requires a header block containing some of these items.

**Usage**

```
cdk.lf(line)
moe.lf(line)
bci.lf(line)
ecfp.lf(line)
fps.lf(line)
jchem.binary.lf(line)
```

**Arguments**

line            The line to parse

**Value**

A list with three components - the name associated with the fingerprint (if available) and a vector of integers representing bits set to 1 (for the case of the first three methods) or a vector of characters representing hashed features (characteristic of circular fingerprints) or more generally, any string feature. The third component is a (possibly empty) list, which contains the remaining components of a line, when the format allows items other than an a title and the fingerprint (such as the FPS format). The content of the third component is dependent on the line function that is being used.

**Author(s)**

Rajarshi Guha <rajarshi.guha@gmail.com>

---

count-methods

*Get or Set Count of Occurrence of a Feature*

---

**Description**

Get or set the count of occurrence associated with a [feature-class](#) object. The default value for the getter (as defined in the prototype) is 1.

**Usage**

```
## S4 method for signature 'feature'
count(object)
## S4 replacement method for signature 'feature,numeric'
count(x) <- value
```

**Arguments**

object            An object of class [feature-class](#)  
x                    An object of class [feature-class](#)  
value              A numeric (which will be coerced to integer) indicating the count associated with the feature

**Value**

An integer representing count of occurrence of the feature

**Methods**

`signature(object = "feature")` Return the count associated with the feature object

`signature(x = "feature", value = "numeric")` Set the count associated with the feature object

**Author(s)**

Rajarshi Guha <rajarshi.guha@gmail.com>

---

distance-methods

*Calculates the Similarity or Dissimilarity Between Two Fingerprints*

---

**Description**

A number of distance metrics can be calculated for binary fingerprints. Some of these are actually similarity metrics and thus represent the reverse of a distance metric.

The following are distance (dissimilarity) metrics

- Hamming
- Mean Hamming
- Soergel
- Pattern Difference
- Variance
- Size
- Shape

The following metrics are similarity metrics and so the distance can be obtained by subtracting the value from 1.0

- Tanimoto
- Dice
- Modified Tanimoto
- Simple
- Jaccard
- Russel-Rao
- Rodgers Tanimoto
- Cosine
- Achiai
- Carbo

- Baroniurbanibuser
- Kulczynski2
- Robust

Finally the method also provides a set of composite and asymmetric distance metrics

- Hamann
- Yule
- Pearson
- Dispersion
- McConnaughey
- Stiles
- Simpson
- Petke
- Tversky

The default metric is the Tanimoto coefficient.

### Usage

```
distance(fp1, fp2, method, a, b)
```

### Arguments

fp1	An object of class fingerprint or featvec
fp2	An object of class fingerprint or featvec
a	Parameter for the Tversky index
b	Parameter for the Tversky index
method	The type of distance metric desired. Partial matching is supported and the default is <code>tanimoto</code> . Alternative values are <ul style="list-style-type: none"><li>• <code>euclidean</code></li><li>• <code>hamming</code></li><li>• <code>meanHamming</code></li><li>• <code>soergel</code></li><li>• <code>patternDifference</code></li><li>• <code>variance</code></li><li>• <code>size</code></li><li>• <code>shape</code></li><li>• <code>jaccard</code></li><li>• <code>dice</code></li><li>• <code>mt</code></li><li>• <code>simple</code></li><li>• <code>russelrao</code></li></ul>

- rodgerstanimoto
- cosine
- achiai
- carbo
- baroniurbanibuser
- kulczynski2
- robust
- hamann
- yule
- pearson
- mcconnaughey
- stiles
- simpson
- petke
- tversky

If the two fingerprints are of class `featvec` then the following methods may be specified: `tanimoto`, `robust` and `dice`.

### Value

Numeric value representing the distance in the specified metric between the supplied fingerprint objects

### Methods

`signature(fp1 = "featvec", fp2 = "featvec", method = "character", a = "missing", b = "missing")`  
Similarity method for feature vector type fingerprints, supporting `tanimoto`, `robust` and `dice` metrics.

`signature(fp1 = "featvec", fp2 = "featvec", method = "missing", a = "missing", b = "missing")`  
Evaluate Tanimoto similarity between two feature vector fingerprints

`signature(fp1 = "fingerprint", fp2 = "fingerprint", method = "character", a = "missing", b = "missing")`  
Evaluate similarity (or dissimilarity) between two binary fingerprints. See below for a list of possible similarity (or dissimilarity) metrics

`signature(fp1 = "fingerprint", fp2 = "fingerprint", method = "character", a = "numeric", b = "numeric")`  
Evaluate Tversky similarity between two binary fingerprints.

`signature(fp1 = "fingerprint", fp2 = "fingerprint", method = "missing", a = "missing", b = "missing")`  
Evaluate Tanimoto similarity between two binary fingerprints

### Author(s)

Rajarshi Guha <rajarshi.guha@gmail.com>

## References

Fligner, M.A.; Verducci, J.S.; Blower, P.E.; A Modification of the Jaccard-Tanimoto Similarity Index for Diverse Selection of Chemical Compounds Using Binary Strings, *Technometrics*, 2002, 44(2), 110-119

Monve, V.; Introduction to Similarity Searching in Chemistry, *MATCH - Comm. Math. Comp. Chem.*, 2004, 51, 7-38

## Examples

```
# make a 2 fingerprint vectors
fp1 <- new("fingerprint", nbit=6, bits=c(1,2,5,6))
fp2 <- new("fingerprint", nbit=6, bits=c(1,2,5,6))

# calculate the tanimoto coefficient
distance(fp1,fp2) # should be 1

# Invert the second fingerprint
fp3 <- !fp2

distance(fp1,fp3) # should be 0
```

---

euc.vector

*Euclidean Representation of Binary Fingerprints*

---

## Description

Ordinarily, a binary fingerprint can be considered to represent a corner of a nD hypercube. However in many cases using such a representation can lead to a very sparse space. Consequently one approach is to convert the fingerprint so that it represents points on a nD unit hypersphere.

The resultant fingerprint is then a nD coordinate.

## Usage

```
euc.vector(fp)
```

## Arguments

fp                    An object of class fingerprint.

## Value

A numeric of length equal to the bit length of the fingerprint. The result corresponds to a unit vector for a point on the nD hypersphere

## Author(s)

Rajarshi Guha <rguha@indiana.edu>

## Examples

```
# make a fingerprint vector
fp <- new("fingerprint", nbit=8, bits=c(1,3,4,5,7))
vec <- euc.vector(fp)
```

---

feature-class	Class "feature"
---------------	-----------------

---

## Description

This class represents features - arbitrary alphanumeric sequences that are used to characterize molecular substructures (though there is no real restriction to molecules). A feature is associated with an integer count, indicating the occurrence of that feature in a molecule. The default value is 1.

## Objects from the Class

Objects can be created by calls of the form `new("feature", ...)`.

## Slots

**feature:** Object of class "character" ~ The string representation of a feature

**count:** Object of class "integer" ~ The occurrence of the feature. Default is 1

**.Data:** ???

## Methods

**count** signature(object = "feature"): Return the count associated with the feature

## Author(s)

Rajarshi Guha <rajarshi.guha@gmail.com>

## See Also

[featvec-class](#)

## Examples

```
## create a new feature
f <- new("feature", feature='ABCD', count=as.integer(1))

## modify the feature string and the count
feature(f) <- 'WXYZ'
count(f) <- 10
```

---

feature-methods	<i>Get or Set the Character String Representing the Feature</i>
-----------------	---

---

**Description**

Get or set the character string representing a feature of a `feature-class` object. The default value for the getter (as defined in the prototype) is the empty string.

**Usage**

```
## S4 method for signature 'feature'
feature(object)
## S4 replacement method for signature 'feature,character'
feature(x) <- value
```

**Arguments**

object	An object of class <code>feature-class</code>
x	An object of class <code>feature-class</code>
value	The character string to replace the current feature string with

**Value**

An character string representing the feature

**Methods**

```
signature(object = "feature") Return the feature associated with the feature object
signature(x = "feature", value = "character") Set the feature associated with the feature
object
```

**Author(s)**

Rajarshi Guha <rajarshi.guha@gmail.com>

---

featvec-class	<i>Class "featvec"</i>
---------------	------------------------

---

**Description**

This class represents feature vector style fingerprints, where, rather than a bit string, the fingerprint is represented as a sequence of (signed) integers or strings. Each element of the collection is a representation of a structural feature. For cases where the features are integers, this usually corresponds to a hash of the original feature string.

### Objects from the Class

Objects can be created by calls of the form `new("featvec", ...)`. In contrast to traditional binary fingerprints, operations on feature vectors are slightly different and essentially correspond to operations on sets. Thus the logical and (&) would correspond to the union of the two feature vectors.

### Slots

**features:** Object of class "character" ~~ A vector containing the numeric or character features. Numeric features are treated as character strings

**provider:** Object of class "character" ~~ Indicates the source of the fingerprint. Can be useful to keep track of what software generated the fingerprint.

**name:** Object of class "character" ~~ The name associated with the fingerprint. If not name is available this gets set to an empty string

**misc:** A list to hold arbitrary items associated with a fingerprint (such as extra fields from a fingerprint file)

### Methods

**distance** signature(fp1 = "featvec", fp2 = "featvec", method = "missing"): ...

**distance** signature(fp1 = "featvec", fp2 = "featvec", method = "character"): ...

**as.character** signature(fp = "featvec"): ...

**length** signature(fp = "featvec"): ...

**show** signature(fp = "featvec"): ...

### Author(s)

Rajarshi Guha <rajarshi.guha@gmail.com>

### See Also

[fp.read](#), [fp.read.to.matrix](#) [fp.sim.matrix](#), [fp.to.matrix](#), [fp.factor.matrix](#) [random.fingerprint](#)

---

fingerprint-class      *Class "fingerprint"*

---

### Description

This class represents binary fingerprints, usually generated by a variety of cheminformatics software, but not restricted to such

## Objects from the Class

Objects can be created by calls of the form `new("fingerprint", ...)`. Fingerprints can traditionally thought of as a vector of 1's and 0's. However for large fingerprints this is inefficient and instead we simply store the positions of the bits that are on. Certain operations also need to know the length of the original bit string and this length is stored in the object at construction. Even though we store extra information along with the bit positions, conceptually we still consider the objects as simple bit strings. Thus the usual bitwise logical operations (&, |, !, xor) can be applied to objects of this class.

## Slots

**bits:** Object of class "numeric" ~~ A vector indicating the bit positions that are on.  
**nbit:** Object of class "numeric" ~~ Indicates the length of the original bit string.  
**folded:** Object of class "logical" ~~ Indicates whether the fingerprint has been folded.  
**provider:** Object of class "character" ~~ Indicates the source of the fingerprint. Can be useful to keep track of what software generated the fingerprint.  
**name:** Object of class "character" ~~ The name associated with the fingerprint. If not name is available this gets set to an empty string  
**misc:** Object of class "list" ~~ A holder for arbitrary items that may have been stored along with the fingerprint. Only certain formats allow extra items to be stored with the fingerprint, so in many cases this field is just an empty list

## Methods

**distance** signature(fp1 = "fingerprint", fp2 = "fingerprint", method = "missing", a = "missing", b = "missing"): ...  
**distance** signature(fp1 = "fingerprint", fp2 = "fingerprint", method = "character", a = "missing", b = "missing"): ...  
**euc.vector** signature(fp = "fingerprint"): ...  
**fold** signature(fp = "fingerprint"): ...  
**random.fingerprint** signature(nbit = "numeric", on = "numeric"): ...

## Author(s)

Rajarshi Guha <rajarshi.guha@gmail.com>

## See Also

[fp.read](#), [fp.read.to.matrix](#) [fp.sim.matrix](#), [fp.to.matrix](#), [fp.factor.matrix](#) [random.fingerprint](#)

## Examples

```
## make fingerprints
x <- new("fingerprint", nbit=128, bits=sample(1:128, 100))
y <- x
distance(x,y) # should be 1
x <- new("fingerprint", nbit=128, bits=sample(1:128, 100))
```

```
distance(x,y)
folded <- fold(x)

## binary operations on fingerprints
x <- new("fingerprint", nbit=8, bits=c(1,2,3,6,8))
y <- new("fingerprint", nbit=8, bits=c(1,2,4,5,7,8))
x & y
x | y
!x
```

---

fold

*Fold a fingerprint*

---

### Description

In many situations a fingerprint is generated using a large length (such as 1024 bits or more). As a result of this, the fingerprints for a dataset can be very sparse. One approach to increasing bit density of such fingerprints is to fold them. This is performed by dividing the original fingerprint bitstring into two substrings of equal length and then perform an OR on the two substrings.

It should be noted that many fingerprint generating routines will perform this internally.

### Usage

```
fold(fp)
```

### Arguments

fp                   The fingerprint to fold. Should be of class fingerprint.

### Value

An object of class fingerprint representing the folded fingerprint.

### Author(s)

Rajarshi Guha <rguha@indiana.edu>

### Examples

```
# make a fingerprint vector
fp <- new("fingerprint", nbit=64, bits=sample(1:64, 30))
fold(fp)
```

---

fp.factor.matrix	<i>Converts a List of Fingerprints to a data.frame of Factors</i>
------------------	---

---

### Description

This function will convert a list of fingerprint objects to a data.frame of factors with levels 1 and 0.

### Usage

```
fp.factor.matrix(fplist)
```

### Arguments

**fplist** A list structure with each element being an object of class fingerprint. These will can be constructed by hand or read from disk via [fp.read](#)

### Value

A matrix with dimensions equal to (length(fplist), length(fplist))

### Author(s)

Rajarshi Guha <rguha@indiana.edu>

### See Also

[distance](#), [fp.read](#)

### Examples

```
# make fingerprint objects
fp1 <- new("fingerprint", nbit=6, bits=c(1,2,5,6))
fp2 <- new("fingerprint", nbit=6, bits=c(1,4,5,6))
fp3 <- new("fingerprint", nbit=6, bits=c(2,3,4,5,6))

fp.factor.matrix( list(fp1,fp2,fp3) )
```

---

fp.read, fp.read.to.matrix

*Functions to Read Fingerprints From Files*

---

## Description

fp.read reads in a set of fingerprints from a file. Fingerprint output from the CDK, MOE and BCI can be handled.

Each fingerprint is represented as a fingerprint object. fp.read returns a list structure, each element being a fingerprint or nfeatvec object, depending on the value of the binary argument.

fp.read.to.matrix is a utility function that reads the fingerprints directly to matrix form (columns are the bit positions and the rows are the objects whose fingerprints have been evaluated). Note that this method does not currently work with feature vector fingerprints.

## Usage

```
fp.read(f='fingerprint.txt', size=1024, lf=cdk.lf, header=FALSE, binary=TRUE)
fp.read.to.matrix(f='fingerprint.txt', size=1024, lf=cdk.lf, header=FALSE)
```

## Arguments

f	File containing the fingerprints
size	The bit length of the fingerprints being considered
lf	A line reading function that parses a single line from a fingerprint file. A number of functions are provided that parse the fingerprints from the output of the CDK, MOE and the BCI toolkit. In addition, support is now available for the FPS format from the chemfp project ( <a href="https://code.google.com/archive/p/chem-fingerprints">https://code.google.com/archive/p/chem-fingerprints</a> ).
header	Indicates whether the first line of the fingerprint file is a header line
binary	If TRUE indicates that a binary fingerprint will be read in. Otherwise indicates that a feature vector style fingerprint (such as from a circular fingerprint) is being read in

## Value

A list or matrix of fingerprints

## Author(s)

Rajarshi Guha <rajarshi.guha@gmail.com>

## See Also

[cdk.lf](#), [moe.lf](#), [bci.lf](#), [ecfp.lf](#), [fps.lf](#)

---

fp.sim.matrix                      *Calculates a Similarity Matrix for a Set of Fingerprints*

---

### Description

Given a set of fingerprints, a pairwise similarity can be calculated using the various distance metrics defined for binary strings. This function calculates the pairwise similarity matrix for a set of fingerprint or featvec objects supplied in a list structure. Any of the distance metrics provided by [distance](#) can be used and the default is the Tanimoto metric.

Note that if the the Euclidean distance is specified then the resultant matrix is a distance matrix and not a similarity matrix

### Usage

```
fp.sim.matrix(fplist, fplist2=NULL, method='tanimoto')
```

### Arguments

fplist	A list structure with each element being an object of class fingerprint or featvec. These can be constructed by hand or read from disk via <a href="#">fp.read</a>
fplist2	A list structure with each element being an object of class fingerprint or featvec. if NULL then traditional pairwise similarity is calculated with each member in fplist, otherwise the resultant N x M matrix is derived from the similarity between each member of fplist and fplist2
method	The type of distance metric to use. The default is tanimoto. Partial matching is supported.

### Value

A matrix with dimensions equal to (length(fplist), length(fplist)) if fplist2 is NULL, otherwise (length(fplist), length(fplist2))

### Author(s)

Rajarshi Guha <rajarshi.guha@gmail.com>

### See Also

[distance](#), [fp.read](#)

### Examples

```
# make fingerprint objects
fp1 <- new("fingerprint", nbit=6, bits=c(1,2,5,6))
fp2 <- new("fingerprint", nbit=6, bits=c(1,4,5,6))
fp3 <- new("fingerprint", nbit=6, bits=c(2,3,4,5,6))

fp.sim.matrix( list(fp1,fp2,fp3) )
```

---

`fp.to.matrix`*Converts a List of Fingerprints to a Matrix*

---

### Description

In general, fingerprint data is read from a file or obtained via calls to an external generator and the return value is a list of fingerprints. This function takes the list and returns a matrix having number of rows equal to the number of fingerprints and the number of columns equal to the length of the fingerprint. Each element is 1 or 0 (1's being specified by the positions in each fingerprint vector)

### Usage

```
fp.to.matrix(fplist)
```

### Arguments

`fplist` A list structure with each element being an object of class fingerprint. These will can be constructed by hand or read from disk via [fp.read](#)

### Value

A matrix with dimensions equal to `length(fplist)`, `bit length`) where bit length is a property of the fingerprint objects in the list.

### Author(s)

Rajarshi Guha <[rguha@indiana.edu](mailto:rguha@indiana.edu)>

### See Also

[distance](#), [fp.read](#)

### Examples

```
# make fingerprint objects
fp1 <- new("fingerprint", nbit=6, bits=c(1,2,5,6))
fp2 <- new("fingerprint", nbit=6, bits=c(1,4,5,6))
fp3 <- new("fingerprint", nbit=6, bits=c(2,3,4,5,6))

fp.to.matrix( list(fp1,fp2,fp3) )
```

---

fplogical

*Logical Operators for Fingerprints*

---

### Description

These functions perform logical operations (AND, OR, NOT, XOR) on the supplied binary fingerprints. Thus for two fingerprints A and B we have

& Logical AND

| Logical OR

xor Logical XOR

! Logical NOT (negation)

### Arguments

e1            An object of class fingerprint

e2            An object of class fingerprint

### Value

A fingerprint object

### Author(s)

Rajarshi Guha <rguha@indiana.edu>

---

length

*Fingerprint Bit Length*

---

### Description

Returns the length of the fingerprint. That is, this is the length of the entire bit string and not simply the number of bits that are on.

### Usage

```
## S4 method for signature 'fingerprint'  
length(x)
```

### Arguments

x            An object of class fingerprint

### Value

The length of the bit string

**Author(s)**

Rajarshi Guha <rguha@indiana.edu>

---

random.fingerprint      *Generate Randomized Fingerprints*

---

**Description**

A utility function that can be used to generate binary fingerprints of a specified length with a specified number of bit positions (selected randomly) set to 1. Currently bit positions are selected uniformly

**Usage**

```
random.fingerprint(nbit,on)
```

**Arguments**

nbit	The length of the fingerprint, that is, the total number of bits. Must be a positive integer.
on	How many positions should be set to 1

**Value**

An object of class fingerprint

**Author(s)**

Rajarshi Guha <rguha@indiana.edu>

**Examples**

```
# make a fingerprint vector
fp <- random.fingerprint(32, 16)
as.character(fp)
```

---

shannon

*Evaluate Shannon Entropy for a Set of Fingerprints*

---

### Description

This method evaluates the Shannon entropy for a set of fingerprints and utilizes the [bit.spectrum](#) method to obtain the relative frequencies of individual bits

### Usage

```
shannon(fplist)
```

### Arguments

**fplist** A list structure with each element being an object of class fingerprint. These will can be constructed by hand or read from disk via [fp.read](#). All fingerprints in the list should be of the same length.

### Value

The Shannon entropy for the set of fingerprints

### Author(s)

Rajarshi Guha <rajarshi.guha@gmail.com>

### See Also

[bit.spectrum](#), [fp.read](#)

---

show

*String Representation of a Fingerprint or Feature*

---

### Description

Simply summarize the fingerprint or feature

### Usage

```
## S4 method for signature 'fingerprint'  
show(object)  
## S4 method for signature 'featvec'  
show(object)  
## S4 method for signature 'feature'  
show(object)
```

**Arguments**

object            An object of class fingerprint, featvec or feature

**Author(s)**

Rajarshi Guha <rajarshi.guha@gmail.com>

# Index

- ! (fplogical), 21
- !, fingerprint-method (fplogical), 21
- \* **classes**
  - feature-class, 12
  - featvec-class, 13
  - fingerprint-class, 14
- \* **logic**
  - as.character, 2
  - c, 6
  - cdk.lf, moe.lf, bci.lf, 6
  - distance-methods, 8
  - euc.vector, 11
  - feature-class, 12
  - featvec-class, 13
  - fingerprint-class, 14
  - fold, 16
  - fp.factor.matrix, 17
  - fp.read, fp.read.to.matrix, 18
  - fp.sim.matrix, 19
  - fp.to.matrix, 20
  - fplogical, 21
  - length, 21
  - random.fingerprint, 22
  - show, 23
- \* **methods**
  - as.character, 2
  - fplogical, 21
  - length, 21
- \* **programming**
  - balance, 3
  - bit.importance, 4
  - bit.spectrum, 5
  - count-methods, 7
  - feature-methods, 13
  - shannon, 23
- & (fplogical), 21
- &, fingerprint, fingerprint-method (fplogical), 21
- as.character, 2
  - as.character, feature-method (as.character), 2
  - as.character, featvec-method (as.character), 2
  - as.character, fingerprint-method (as.character), 2
- balance, 3
- bci.lf, 18
- bci.lf (cdk.lf, moe.lf, bci.lf), 6
- bit.importance, 3, 4
- bit.spectrum, 3, 4, 5, 23
- c, 6
- c, feature-method (c), 6
- cdk.lf, 18
- cdk.lf (cdk.lf, moe.lf, bci.lf), 6
- cdk.lf, moe.lf, bci.lf, 6
- count (count-methods), 7
- count, feature-method (count-methods), 7
- count-methods, 7
- count<- (count-methods), 7
- count<-, feature, numeric-method (count-methods), 7
- count<--methods (count-methods), 7
- distance, 5, 17, 19, 20
- distance (distance-methods), 8
- distance, featvec, featvec, character, missing, missing-method (distance-methods), 8
- distance, featvec, featvec, character-method (featvec-class), 13
- distance, featvec, featvec, missing, missing, missing-method (distance-methods), 8
- distance, featvec, featvec, missing-method (featvec-class), 13
- distance, fingerprint, fingerprint, character, missing, missing-method (distance-methods), 8
- distance, fingerprint, fingerprint, character, numeric, numeric-method (distance-methods), 8

distance, fingerprint, fingerprint, missing, missing, missing-method  
     (distance-methods), 8  
 distance-methods, 8  
 ecfp.lf, 18  
 ecfp.lf (cdk.lf, moe.lf, bci.lf), 6  
 entropy (shannon), 23  
 euc.vector, 11  
 euc.vector, fingerprint-method  
     (fingerprint-class), 14  
  
 feature (feature-methods), 13  
 feature, feature-method  
     (feature-methods), 13  
 feature-class, 12  
 feature-methods, 13  
 feature<- (feature-methods), 13  
 feature<-, feature, character-method  
     (feature-methods), 13  
 feature<--methods (feature-methods), 13  
 featvec-class, 13  
 fingerprint-class, 14  
 fold, 16  
 fold, fingerprint-method  
     (fingerprint-class), 14  
 fp.factor.matrix, 14, 15, 17  
 fp.read, 5, 14, 15, 17, 19, 20, 23  
 fp.read (fp.read, fp.read.to.matrix), 18  
 fp.read, fp.read.to.matrix, 18  
 fp.read.to.matrix, 14, 15  
 fp.sim.matrix, 14, 15, 19  
 fp.to.matrix, 14, 15, 20  
 fplogical, 21  
 fps.lf, 18  
 fps.lf (cdk.lf, moe.lf, bci.lf), 6  
  
 jchem.binary.lf (cdk.lf, moe.lf,  
     bci.lf), 6  
  
 length, 21  
 length, featvec-method (featvec-class),  
     13  
 length, fingerprint-method (length), 21  
  
 moe.lf, 18  
 moe.lf (cdk.lf, moe.lf, bci.lf), 6  
  
 random.fingerprint, 14, 15, 22  
 random.fingerprint, numeric, numeric-method  
     (fingerprint-class), 14