|  |  |
| --- | --- |
| match {base} | R Documentation |

**Value Matching**

**Description**

match returns a vector of the positions of (first) matches of its first argument in its second.

%in% is a more intuitive interface as a binary operator, which returns a logical vector indicating if there is a match or not for its left operand.

**Usage**

match(x, table, nomatch = NA\_integer\_, incomparables = NULL)

x %in% table

**Arguments**

|  |  |
| --- | --- |
| x | vector or NULL: the values to be matched. [Long vectors](http://127.0.0.1:14695/library/base/help/Long%20vectors) are supported. |
| table | vector or NULL: the values to be matched against. [Long vectors](http://127.0.0.1:14695/library/base/help/Long%20vectors) are not supported. |
| nomatch | the value to be returned in the case when no match is found. Note that it is coerced to integer. |
| incomparables | a vector of values that cannot be matched. Any value in x matching a value in this vector is assigned the nomatch value. For historical reasons, FALSE is equivalent to NULL. |

**Details**

%in% is currently defined as

"%in%" <- function(x, table) match(x, table, nomatch = 0) > 0

Factors, raw vectors and lists are converted to character vectors, and then x and table are coerced to a common type (the later of the two types in **R**'s ordering, logical < integer < numeric < complex < character) before matching. If incomparables has positive length it is coerced to the common type.

Matching for lists is potentially very slow and best avoided except in simple cases.

Exactly what matches what is to some extent a matter of definition. For all types, NA matches NA and no other value. For real and complex values, NaN values are regarded as matching any other NaN value, but not matching NA.

That %in% never returns NA makes it particularly useful in if conditions.

Character strings will be compared as byte sequences if any input is marked as "bytes" (see [Encoding](http://127.0.0.1:14695/library/base/help/Encoding)).

**Value**

A vector of the same length as x.

match: An integer vector giving the position in table of the first match if there is a match, otherwise nomatch.

If x[i] is found to equal table[j] then the value returned in the i-th position of the return value is j, for the smallest possible j. If no match is found, the value is nomatch.

%in%: A logical vector, indicating if a match was located for each element of x: thus the values are TRUE or FALSE and never NA.

**References**

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

**See Also**

[pmatch](http://127.0.0.1:14695/library/base/help/pmatch) and [charmatch](http://127.0.0.1:14695/library/base/help/charmatch) for (*partial*) string matching, [match.arg](http://127.0.0.1:14695/library/base/help/match.arg), etc for function argument matching. [findInterval](http://127.0.0.1:14695/library/base/help/findInterval) similarly returns a vector of positions, but finds numbers within intervals, rather than exact matches.

[is.element](http://127.0.0.1:14695/library/base/help/is.element) for an S-compatible equivalent of %in%.

**Examples**

## The intersection of two sets can be defined via match():

## Simple version:

## intersect <- function(x, y) y[match(x, y, nomatch = 0)]

intersect # the R function in base is slightly more careful

intersect(1:10, 7:20)

1:10 %in% c(1,3,5,9)

sstr <- c("c","ab","B","bba","c",NA,"@","bla","a","Ba","%")

sstr[sstr %in% c(letters, LETTERS)]

"%w/o%" <- function(x, y) x[!x %in% y] #-- x without y

(1:10) %w/o% c(3,7,12)

## Note that setdiff() is very similar and typically makes more sense:

c(1:6,7:2) %w/o% c(3,7,12) # -> keeps duplicates

setdiff(c(1:6,7:2), c(3,7,12)) # -> unique values