|  |  |
| --- | --- |
| factor {base} | R Documentation |

**Factors**

**Description**

The function factor is used to encode a vector as a factor (the terms ‘category’ and ‘enumerated type’ are also used for factors). If argument ordered is TRUE, the factor levels are assumed to be ordered. For compatibility with S there is also a function ordered.

is.factor, is.ordered, as.factor and as.ordered are the membership and coercion functions for these classes.

**Usage**

factor(x = character(), levels, labels = levels,

exclude = NA, ordered = is.ordered(x), nmax = NA)

ordered(x, ...)

is.factor(x)

is.ordered(x)

as.factor(x)

as.ordered(x)

addNA(x, ifany = FALSE)

**Arguments**

|  |  |
| --- | --- |
| x | a vector of data, usually taking a small number of distinct values. |
| levels | an optional vector of the values (as character strings) that x might have taken. The default is the unique set of values taken by [as.character](http://127.0.0.1:14695/library/base/help/as.character)(x), sorted into increasing order *of x*. Note that this set can be specified as smaller than sort(unique(x)). |
| labels | *either* an optional character vector of labels for the levels (in the same order as levels after removing those in exclude), *or* a character string of length 1. |
| exclude | a vector of values to be excluded when forming the set of levels. This should be of the same type as x, and will be coerced if necessary. |
| ordered | logical flag to determine if the levels should be regarded as ordered (in the order given). |
| nmax | an upper bound on the number of levels; see ‘Details’. |
| ... | (in ordered(.)): any of the above, apart from ordered itself. |
| ifany | (only add an NA level if it is used, i.e. if any(is.na(x)). |

**Details**

The type of the vector x is not restricted; it only must have an [as.character](http://127.0.0.1:14695/library/base/help/as.character) method and be sortable (by [sort.list](http://127.0.0.1:14695/library/base/help/sort.list)).

Ordered factors differ from factors only in their class, but methods and the model-fitting functions treat the two classes quite differently.

The encoding of the vector happens as follows. First all the values in exclude are removed from levels. If x[i] equals levels[j], then the i-th element of the result is j. If no match is found for x[i] in levels (which will happen for excluded values) then the i-th element of the result is set to [NA](http://127.0.0.1:14695/library/base/help/NA).

Normally the ‘levels’ used as an attribute of the result are the reduced set of levels after removing those in exclude, but this can be altered by supplying labels. This should either be a set of new labels for the levels, or a character string, in which case the levels are that character string with a sequence number appended.

factor(x, exclude = NULL) applied to a factor is a no-operation unless there are unused levels: in that case, a factor with the reduced level set is returned. If exclude is used it should also be a factor with the same level set as x or a set of codes for the levels to be excluded.

The codes of a factor may contain [NA](http://127.0.0.1:14695/library/base/help/NA). For a numeric x, set exclude = NULL to make [NA](http://127.0.0.1:14695/library/base/help/NA) an extra level (prints as <NA>); by default, this is the last level.

If NA is a level, the way to set a code to be missing (as opposed to the code of the missing level) is to use [is.na](http://127.0.0.1:14695/library/base/help/is.na) on the left-hand-side of an assignment (as in is.na(f)[i] <- TRUE; indexing inside is.na does not work). Under those circumstances missing values are currently printed as <NA>, i.e., identical to entries of level NA.

is.factor is generic: you can write methods to handle specific classes of objects, see [InternalMethods](http://127.0.0.1:14695/library/base/help/InternalMethods).

Where levels is not supplied, [unique](http://127.0.0.1:14695/library/base/help/unique) is called. Since factors typically have quite a small number of levels, for large vectors x it is helpful to supply nmax as an upper bound on the number of unique values.

**Value**

factor returns an object of class "factor" which has a set of integer codes the length of x with a "levels" attribute of mode [character](http://127.0.0.1:14695/library/base/help/character) and unique (![anyDuplicated](http://127.0.0.1:14695/library/base/help/anyDuplicated)(.)) entries. If argument ordered is true (or ordered() is used) the result has class c("ordered", "factor").

Applying factor to an ordered or unordered factor returns a factor (of the same type) with just the levels which occur: see also [[.factor](http://127.0.0.1:14695/library/base/help/%5B.factor) for a more transparent way to achieve this.

is.factor returns TRUE or FALSE depending on whether its argument is of type factor or not. Correspondingly, is.ordered returns TRUE when its argument is an ordered factor and FALSE otherwise.

as.factor coerces its argument to a factor. It is an abbreviated form of factor.

as.ordered(x) returns x if this is ordered, and ordered(x) otherwise.

addNA modifies a factor by turning NA into an extra level (so that NA values are counted in tables, for instance).

**Warning**

The interpretation of a factor depends on both the codes and the "levels" attribute. Be careful only to compare factors with the same set of levels (in the same order). In particular, as.numeric applied to a factor is meaningless, and may happen by implicit coercion. To transform a factor f to approximately its original numeric values, as.numeric(levels(f))[f] is recommended and slightly more efficient than as.numeric(as.character(f)).

The levels of a factor are by default sorted, but the sort order may well depend on the locale at the time of creation, and should not be assumed to be ASCII.

There are some anomalies associated with factors that have NA as a level. It is suggested to use them sparingly, e.g., only for tabulation purposes.

**Comparison operators and group generic methods**

There are "factor" and "ordered" methods for the [group generic](http://127.0.0.1:14695/library/base/help/group%20generic) [Ops](http://127.0.0.1:14695/library/base/help/S3groupGeneric) which provide methods for the [Comparison](http://127.0.0.1:14695/library/base/help/Comparison) operators, and for the [min](http://127.0.0.1:14695/library/base/help/min), [max](http://127.0.0.1:14695/library/base/help/max), and [range](http://127.0.0.1:14695/library/base/help/range) generics in [Summary](http://127.0.0.1:14695/library/base/help/S3groupGeneric) of "ordered". (The rest of the groups and the [Math](http://127.0.0.1:14695/library/base/help/S3groupGeneric) group generate an error as they are not meaningful for factors.)

Only == and != can be used for factors: a factor can only be compared to another factor with an identical set of levels (not necessarily in the same ordering) or to a character vector. Ordered factors are compared in the same way, but the general dispatch mechanism precludes comparing ordered and unordered factors.

All the comparison operators are available for ordered factors. Collation is done by the levels of the operands: if both operands are ordered factors they must have the same level set.

**Note**

In earlier versions of **R**, storing character data as a factor was more space efficient if there is even a small proportion of repeats. However, identical character strings now share storage, so the difference is small in most cases. (Integer values are stored in 4 bytes whereas each reference to a character string needs a pointer of 4 or 8 bytes.)

**References**

Chambers, J. M. and Hastie, T. J. (1992) *Statistical Models in S*. Wadsworth & Brooks/Cole.

**See Also**

[[.factor](http://127.0.0.1:14695/library/base/help/%5B.factor) for subsetting of factors.

[gl](http://127.0.0.1:14695/library/base/help/gl) for construction of balanced factors and [C](http://127.0.0.1:14695/library/base/help/C) for factors with specified contrasts. [levels](http://127.0.0.1:14695/library/base/help/levels) and [nlevels](http://127.0.0.1:14695/library/base/help/nlevels) for accessing the levels, and [unclass](http://127.0.0.1:14695/library/base/help/unclass) to get integer codes.

**Examples**

(ff <- factor(substring("statistics", 1:10, 1:10), levels = letters))

as.integer(ff) # the internal codes

(f. <- factor(ff)) # drops the levels that do not occur

ff[, drop = TRUE] # the same, more transparently

factor(letters[1:20], labels = "letter")

class(ordered(4:1)) # "ordered", inheriting from "factor"

z <- factor(LETTERS[3:1], ordered = TRUE)

## and "relational" methods work:

stopifnot(sort(z)[c(1,3)] == range(z), min(z) < max(z))

## suppose you want "NA" as a level, and to allow missing values.

(x <- factor(c(1, 2, NA), exclude = NULL))

is.na(x)[2] <- TRUE

x # [1] 1 <NA> <NA>

is.na(x)

# [1] FALSE TRUE FALSE

## Using addNA()

Month <- airquality$Month

table(addNA(Month))

table(addNA(Month, ifany = TRUE))