

R Cheat Sheet: Factors

Factors Factor arithmetic & Boolean comparisons

- A one-dimensional array of categorical - factors cannot be added, multiplied, etc. (unordered) or ordinal (ordered) data. - same-type factors are equality testable
 - Indexed from 1 to N. Not fixed length. `z <- sex.f[1] == sex.f[2] # OKAY`
 - Named factors are possible (but rare) `z <- sex.f[1] == size.f[2] # WRONG`
 Trap: the hidden/unexpected coercion of an - ordered factors can be order compared
object to a factor is a key source of bugs `z <- size.f[1] < size.f[2] # OKAY`
`z <- sex.f[1] < sex.f[2] # WRONG`

Why use factors

1 Specifying a non-alphabetical order Managing the enumeration (levels)

2 Some statistical functions treat cat/ord `f <- factor(letters[1:3]) # example data`
 data differently from continuous data. `levels(f) # -> get all levels`

3 Deep ggplot2 code depends on it `levels(f)[1] # -> get a specific level`

test existence of a level

Create any `(levels(f) %in% c('a', 'b')) # -> TRUE`

Example 1 - unordered add new levels:

`sex.v <- c('M', 'F', 'F', 'M', 'M', 'F') levels(f)[length(levels(f))+1] <- 'ZZ'`

`sex.f <- factor(sex.v) # unordered levels(f) <- c(levels(f), 'AA')`

`sex.w <- as.character(sex.f) # restore reorder levels`

Eg 2 - ordered (small, medium, large) `levels(f) # -> 'a' 'b' 'c' 'ZZ' 'AA'`

`size.v <- c('S', 'L', 'M', 'L', 'S', 'M') f <- factor(f, levels(f)[c(4,1:3,5)])`

`size.f <- factor(size.v, ordered=TRUE) change/rename levels`

ordered L < M < S from underlying type `levels(f)[1] <- 'XX' # rename a level`

Eg 3 - ordered, where we set the order `levels(f)[levels(f) %in% 'AA'] <- 'BB'`

`size.lvls <- c('S', 'M', 'L') # set order delete (or drop) unused levels`

`sz2.f <- factor(size.v, levels=size.lvls) f <- f[drop=TRUE]`

above: ordered (low to high) by levels

Eg 4 - ordered with levels and labels Adding an element to a factor

`levels <- c(1, 2, 3, 99) # from codesheet f <- factor(letters[1:10]) # example data`

`labels <- c('Love', 'Neutral', 'Hate', NA) f[length(f) + 1] <- 'a' # add at end`

`data.v <- c(1, 2, 3, 99, 1, 2, 1, 2, 99) Trap: above only adds an existing level`

`data.f <- factor(data.v, levels=levels, Tip: decode/recode for general add below`

`labels=labels) f <- factor(c(as.character(f), 'zz'))`

levels: input - how factor() reads in

labels: output - how factor() puts out Merging/combining factors

Note: if specified, labels become `a <- factor(1:10); b <- factor(letters[a])`

the internal reference and coding frame `union <- factor(c(as.character(a),`

`Eg 5 - using the cut function to group as.character(b))) # union`

`i <- 1:50 + rnorm(50,0,5); k <- cut(i, 5) cross <- interaction(a, b) # a.b`

both merges produced unordered factors

Basic information about a factor # Levels: union 20; cross 100

Function Returns # Items: union 20; cross 10.

`dim(f) NULL`

`is.factor(f) TRUE` Using factors within data frames

`is.atomic(f) TRUE # df$x <- reorder(df$f, df$x, F, order=T)`

`is.vector(f) FALSE # yields factor ordered by function F`

`is.list(f) FALSE # applied to col X grouped by col f`

`is.recursive(f) FALSE # by(df$x, df$f, F) - apply F by factor f`

`length(f) Non-negative number`

`names(f) NULL or char vector Traps`

`mode(f) "numeric" 1 Strings loaded from a file converted`

`class(f) "factor" to factors (Hint: in read.table or`

`typeof(f) "integer" read.csv use: stringsAsFactors=FALSE)`

`is.ordered(f) TRUE or FALSE 2 Numbers from a file factorised. Revert:`

`unclass(f) # -> R's internal coding as.numeric(levels(f))[as.integer(f)]`

`cat(f); print(f); str(f); dput(f); head(f) 3 One factor (enumeration) cannot be`
 meaningfully compared with another.

Indexing: much like atomic vectors 4 NA's (missing data) in factors and

- `[x]` selects a factor for the cell/range x levels can cause problems (Hint: avoid)

- `[[x]]` selects a length=1 factor for the 5 Adding a row to a data frame, which
 single cell index x (rarely used) adds a new level to a column factor.

- The \$ operator is invalid with factors (Hint: make the new row a data frame
 with a factor column then use `rbind()`).