

R Cheat Sheet: Factors

Factors

```
# A one-dimensional array of categorical
# (unordered) or ordinal (ordered) data,
# indexed from 1 to N. Not fixed length.
```

Why use factors

```
# 1) specifying a non-alphabetical order
# 2) because some statistical functions
#    treat cat/ord data differently to
#    continuous data,
# 3) deep ggplot2 code depends on it
```

Create

```
# Example 1 - unordered
sex.v <- c('M', 'F', 'F', 'M', 'M', 'F')
sex.f <- factor(sex.v) # unordered
sex.w <- as.character(sex.f) # convert back
```

```
# Eg 2 - ordered (small, medium, large)
size.v <- c('S', 'L', 'M', 'L', 'S', 'M')
size1.f <- factor(size.v, ordered=TRUE)
# ordered L < M < S from underlying type
```

```
# Eg 3 - ordered, where we set the order
size.lvls <- c('S', 'M', 'L') # set order
size2.f <- factor(size.v, levels=size.lvls)
# above: ordered (low to high) by levels
```

```
# Eg 4 - ordered with levels and labels
levels <- c(1, 2, 3, 99) # from code sheet
labels <- c('Love', 'Neutral', 'Hate', NA)
data.v <- c(1, 2, 3, 99, 1, 2, 1, 2, 99)
data.f <- factor(data.v, levels=levels,
                 labels=labels)
# levels: input - how factor() reads in
# labels: output - how factor() puts out
# Note: if labels are specified they become
# the internal reference and coding frame
```

```
# Eg 5 - using the cut function to group
i <- 1:50 + rnorm(50, 0, 5); k <- cut(i, 5)
```

Basic information about a factor

```
is.factor(sex.f) # -> TRUE
is.ordered(sex.f) # -> FALSE
mode(sex.f) # -> "numeric" # always!
class(sex.f) # -> "factor" # always!
typeof(sex.f) # -> "integer" # always!
unclass(sex.f) # -> R's internal coding
# Also: attributes(sex.f); length(sex.f)
# table(sex.f); head(sex.f); tail(sex.f)
# str(sex.f); dput(sex.f); summary(sex.f)
```

Indexing [k]

```
# numerically indexed - not by name
x <- sex.f[1] # get the first factor
y <- sex.f[length(sex.f)] # last one
sex.f[1] <- '?' # change specific factor
sex.f[1:5] <- 'M' # change multiple
sex.f[1:6] <- c('M', 'F', 'F', 'M',
               'M', 'F') # refresh the factor
# Note: sex.f <- c('M', 'F', 'F', 'M',
#                  'M', 'F') puts a vector in sex.f
```

Factor arithmetic & Boolean comparisons

```
# factors cannot be added, multiplied, etc.
# same-type factors are equality testable
z <- sex.f[1] == sex.f[2] # OKAY
z <- sex.f[1] == size.f[2] # WRONG
# ordered factors can be order compared
z <- size1.f[1] < size1.f[2] # OKAY
z <- sex.f[1] < sex.f[2] # WRONG
```

Managing the enumeration (levels)

```
f <- factor(letters[1:3]) # example data
levels(f) # -> get all levels
levels(f)[1] # -> get a specific level
# test existence of a level
any(levels(f) %in% 'b') # -> TRUE
# add new levels:
levels(f)[length(levels(f))+1] <- 'ZZ'
levels(f) <- c(levels(f), 'AA')
# reorder levels
levels(f) # -> 'a' 'b' 'c' 'ZZ' 'AA'
f <- factor(f, levels(f)[c(4,1:3,5)])
# change/rename levels
levels(f)[1] <- 'XX' # rename a level
levels(f)[levels(f) %in% 'AA'] <- 'BB'
# delete (or drop) unused levels
f <- f[drop=TRUE]
```

Adding an element to a factor

```
f <- factor(letters[1:10]) # example data
f[length(f) + 1] <- 'a' # add at end
# Trap: above only adds an existing level
# Tip: decode/recode for general add below
f <- factor(c(as.character(f), 'jj'))
```

Merging/combining factors

```
a <- factor(1:10); b <- factor(letters[a])
union <- factor(c(as.character(a),
                  as.character(b))) # union
cross <- interaction(a, b) # a.b
# both merges produced unordered factors
# Levels: union 20; cross 100
# Items: union 20; cross 10.
```

Using factors within data frames

```
# df$x <- reorder(df$f, df$X, F, order=T)
# yields factor ordered by function F
# applied to col X grouped by col f
# by(df$x, df$f, F) - apply F by factor f
```

Traps

```
# 1) Strings loaded from a file converted
#    to factors (Hint: in read.table or
#    read.csv use: stringsAsFactors=FALSE)
# 2) Numbers from a file factorised. Revert
#    as.numeric(levels(f))[as.integer(f)]
# 3) One factor (enumeration) cannot be
#    meaningfully compared with another.
# 4) NA's (missing data) in factors and
#    levels can cause problems (Hint:avoid)
# 5) Adding a row to a data frame, which
#    adds a new level to a column factor.
#    (Hint: make the new row a data frame
#    with a factor column then use rbind).
```