R Cheat Sheet: Atomic Vectors (often just called "vectors" in R)

Atomic vectors: The contents of a vector
- An object with contiguous, indexed values cat(v); print(v) # print vector contents
- Indexed from 1 to length(vector) str(v); dput(v); # print vector structure
- All values of the same basic atomic type head(v); tail(v) # first/last items in v
- Vectors do not have a dimension attribute
- Has a fixed length once created Indexing: [and [[(but not $)
- [x] selects a vector for the cell/range x

Six basic atomic types: - [[x]] selects a length=1 vector for the

Class	Example	single cell index x (*rarely used*)
logical	TRUE, FALSE, NA	- $ operator invalid for atomic vectors
integer	1:5, 2L, 4L, 6L	Index by positive numbers: *these ones*
numeric	2, 0.77 (double precision)	v[c(1,1,4)] # get 1st one twice then 4th
complex	3.7+4.2i, 0+1i	v[m:n] # get elements from indexes m to n
character	"string", 'another string'	v[[7]] <- 6 # set seventh element to 6
raw	(byte data from 0-255)	v[which(v == 'M')] # which() yields nums

Index by negative numbers: *not these*

No scalars v[-1] # get all but the first element

In R, these basic types are always in a v[-length(v)] # get all but the last one

vector. Scalars are just length=1 vectors. v[-c(1,3,5,7,9)] # get all but ...

Index by logical atomic vector: *in/out*

Creation (length determined at creation) v[c(TRUE, FALSE, TRUE)] # get 1st and 3rd

Default value vectors of length=4 v[v > 2] # get all where v is g.t. two

u <- vector(mode='logical', length=4) v[v > 2 & v < 9] # get where v>2 and v<9

print(u) # -> FALSE, FALSE, FALSE, FALSE v[v == 'M'] # get where v equals char 'M'

v <- vector(mode='integer', length=4) v[v %in% c('me', 'andMe', 'meToo')] # get

Also: numeric(4); character(4); raw(4) Indexed by name (*only with named vectors*)

Using the sequence operator v[['alpha']] # get single by name

i <- 1:5 # produces an integer sequence v[['beta']] <- 'b' # set single by name

j <- 1.4:6.4 # a numeric sequence v[c('alpha', 'beta')] # get multiple

k <- seq(from=0, to=1, by=0.1) # numeric v[!(names(v) %in% c('a', 'b'))] # exclude

Using the c() function names(v)['z'] <- 'omega' # change name

l <- c(TRUE, FALSE) # logical vector

n <- c(1.3, 7, 7/20) # numeric vector Most functions/operators are vectorised

z <- c(1+2i, 2, -3+4i) # complex vector c(1,3,5) + c(5,3,1) # -> 6, 6, 6

c <- c('pink', 'blue') # character vector c(1,3,5) * c(5,3,1) # -> 5, 9, 5

Other things

v1 <- c(a=1, b=2, c=3) # a named vector Sorting

v2 <- rep(NA, 3) # 3 repeated NAs upSorted <- sort(v) # also: v[order(v)]

v3 <- c(v1, v2) # concatenate and flatten d <- sort(v, decreasing=TRUE) # rev(sort(v))

v4 <- append(origV, insertV, position)

Raw vectors (byte sequences)

Conversion s <- charToRaw('raw') # string input

as.vector(v); as.logical(v); as.integer(v) r <- as.raw(c(114, 97, 119)) # decimal in

as.numeric(v); as.character(v) # etc. etc. print(r) # -> 72 61 77 (hex output)

unlist(l) # convert list to atomic vector

Trap: unlist() wont unlist non-atomic items Traps

unlist(list(as.name('fred'))) # FAILS Recycling vectors in math operations

c(1,2,3,4,5) + 1 # -> 2, 3, 4, 5, 6

Basic information about atomic vectors c(1,2,3,4,5) * c(1,0) # -> 1, 0, 3, 0, 5

Function	Returns	Automatic type coercion (often hidden)
dim(v)	NULL	x <- c(5, 'a') # c() converts 5 to '5'
is.atomic(v)	TRUE	x <- 1:3; x[3] <- 'a' # x now '1' '2' 'a'
is.vector(v)	TRUE	typeof(1:2) == typeof(c(1,2)) # -> FALSE
is.list(v)	FALSE	For-loops on empty vectors
is.factor(v)	FALSE	for(i in 1:length(c())) print(i) # loopx2
is.recursive(v)	FALSE	for(i in seq_len(x)) # empty vector safe
length(v)	Non-negative number	Also: for(j in seq_along(x))
names(v)	NULL or char vector	Some Boolean ops not vectorised
mode(v); class(v); typeof(v); attributes(v)		c(T,F,T) && c(T,F,F) # TRUE (*!vectorised*)

is.numeric(v); is.character(v); # etc. etc. c(T,F,T) & c(T,F,F) # TRUE, FALSE, FALSE

Trap: lists are vectors (but not atomic) Similarly: || is not vectorised; | is

Trap: array/matrix are atomic (not vectors) Factor indexes are treated as integers

Tip: use (is.vector(v) && is.atomic(v)) Tip: decode with v[as.character(f)].