R Cheat Sheet: Matrices and Arrays

Context Matrix manipulation

Matrices and arrays are an extension on R's newM <- cbind(M, N, ...) # horizontal join
atomic vectors. Quick recap: atomic vectors newM <- rbind(M, N, ...) # vertical join
contain *values* (not objects). They hold a # M and N either matrices or atomic vectors
contiguous set of values, all of which are v <- c(M) # convert matrix back to a vector
of the same basic type. There are six types df <- data.frame(M) # convert to data frame
of atomic vector: logical, integer,
numeric, complex, character and raw. Matrix multiplication
Importantly: atomic vectors have no InnerProduct <- A %*% B # matrix multiply
dimension attribute. Matrices and arrays OuterProduct <- A %o% B
are effectively vectors with a dimension CrossProduct <- crossprod(A, B)
attribute. Matrices are two-dimensional Trap: A * B -> element wise multiplication
(tabular) objects, containing values all of
the same type (unlike data frames). Arrays Matrix maths
are multi-dimensional objects (typically rowMeans(M) # R vector of row means
with three plus dimensions), with values colMeans(M) # R vector of column means
all of the same type. rowSums(M) # R vector of row sums
colSums(M) # R vector of column sums
Matrix versus data.frame t <- t(M) # transpose the M matrix
In a matrix, every column, and every cell inverse <- solve(M) # get the inverse of M
is of the same basic atomic type. In a # solve the system of equations Mx = b
data.frame each column can be of a x <- solve(M, b) # simultaneous equation
different type (eg. numeric, character, e <- eigen(M) # -> list with values/vectors
factor). Data frames are best with messy d <- det(M) # determinant of square matrix
data, and for variables of mixed modes.
Matrix indexing [row, col] [[row, col]]
Matrix creation # [[for single cell selection; [for multi
# generalCase <- matrix(data=NA, nrow=1, # indexed by positive numbers: *these ones*
# ncol=1, byrow=FALSE, dimnames=NULL) # indexed by negative numbers: *not these*
M <- matrix(# indexed by logical atomic vector: *in/out*
c(2, -1, 5, -1, 2, -1, 9, -3, 4), # named rows/cols can be indexed by name
nrow=3, ncol=3, byrow=TRUE) # M[i] or M[[i]] is vector-like indexing
# which yields the following 3x3 matrix: # $ operator is invalid for atomic vectors
# [,1] [,2] [,3] # M[r,] # get/set selected row(s)
# [1,] 2 -1 5 # M[,c] # get/set selected col(s)
# [2,] -1 2 -1
# [3,] 9 -3 4 Arrays
# Trap: R vectors are not matrix column A <- array(1:8, dim=c(2,2,2))
# vectors; however, the matrix class # A three dimensional example
# produces 1-column vectors by default # , , 1
b <- matrix(c(0, -1, 4)) # column vector # [,1] [,2]
I <- diag(3) # create a 3x3 identity matrix # [1,] 1 3
D <- diag(c(1,2,3)) # 3x3 with speced diag # [2,] 2 4
d <- diag(M) # R vector with the diag of M # , , 2
MDF <- as.matrix(df) # data.frame to matrix # [,1] [,2]
# [1,] 5 7
Basic information about a matrix # [2,] 6 8
Function Returns # Could have created in two steps:
dim(M) NROW NCOL (2 numbers) A <- 1:8; dim(A) <- c(2,2,2)
class(M) "matrix" # A matrix is a special case of array ...
is.matrix(M) TRUE M <- array(1:9, dim=c(3,3)) # a matrix
is.array(M) TRUE # Matrices are arrays with two dimensions
is.atomic(M) TRUE
is.vector(M) FALSE
is.list(M) FALSE
is.factor(M) FALSE
is.recursive(M) FALSE
nrow(M); ncol(M) Row and Col counts
length(M) NROW*NCOL (1 number)
rownames(M) NULL or char vector
colnames(M) NULL or char vector