R Cheat Sheet: Matrices and Arrays

Context

Matrices and arrays are an extension on R's atomic vectors. Quick recap: atomic vectors contain *values* (not objects). They hold a contiguous set of values, all of which are of the same basic type. There are six types of atomic vector: logical, integer, numeric, complex, character and raw. Importantly: atomic vectors have no dimension attribute. Matrices and arrays are effectively vectors with a dimension attribute. Matrices are two-dimensional (tabular) objects, containing values all of the same type (unlike data frames). Arrays are multi-dimensional objects (typically with three plus dimensions), with values all of the same type.

Matrix versus data.frame

In a matrix, every column, and every cell is of the same basic atomic type. In a data.frame each column can be of a different type (eg. numeric, character, factor). Data frames are best with messy data, and for variables of mixed modes.

Matrix creation

```
# generalCase <- matrix(data=NA, nrow=1,
#       ncol=1, byrow=FALSE, dimnames=NULL)
M <- matrix(
       c(2, -1, 5, -1, 2, -1, 9, -3, 4),
       nrow=3, ncol=3, byrow=TRUE)
# which yields the following 3x3 matrix:
#         [,1] [,2] [,3]
#    [1,]    2   -1    5
#    [2,]   -1    2   -1
#    [3,]    9   -3    4
# Trap: R vectors are not matrix column
# vectors; however, the matrix class
# produces 1-column vectors by default
b <- matrix(c(0, -1, 4)) # column vector
I <- diag(3) # create a 3x3 identity matrix
D <- diag(c(1,2,3)) # 3x3 with speced diag
d <- diag(M) # R vector with the diag of M
MDF <- as.matrix(df) # data.frame to matrix
```

Basic information about a matrix

Function	Returns
dim(M)	NROW NCOL (2 numbers)
class(M)	"matrix"
is.matrix(M)	TRUE
is.array(M)	TRUE
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

Matrix manipulation

```
newM <- cbind(M, N, ...) # horizontal join
newM <- rbind(M, N, ...) # vertical join
# M and N either matrices or atomic vectors
v <- c(M) # convert matrix back to a vector
df <- data.frame(M) # convert to data frame
```

Matrix multiplication

```
InnerProduct <- A %*% B # matrix multiply
OuterProduct <- A %o% B
CrossProduct <- crossprod(A, B)
Trap: A * B -> element wise multiplication
```

Matrix maths

```
rowMeans(M) # R vector of row means
colMeans(M) # R vector of column means
rowSums(M) # R vector of row sums
colSums(M) # R vector of column sums
t <- t(M) # transpose the M matrix
inverse <- solve(M) # get the inverse of M
# solve the system of equations Mx = b
x <- solve(M, b) # simultaneous equation
e <- eigen(M) # -> list with values/vectors
d <- det(M) # determinant of square matrix
```

Matrix indexing [row, col] [[row, col]]

```
# [[ for single cell selection; [ for multi
# indexed by positive numbers: these ones
# indexed by negative numbers: not these
# indexed by logical atomic vector: in/out
# named rows/cols can be indexed by name
# M[i] or M[[i]] is vector-like indexing
# $ operator is invalid for atomic vectors
# M[r,] # get/set selected row(s)
# M[,c] # get/set selected col(s)
```

Arrays

```
A <- array(1:8, dim=c(2,2,2))
# A three dimensional example
#  , , 1
#  [,1] [,2]
#  [1,]    1    3
#  [2,]    2    4
#  , , 2
#  [,1] [,2]
#  [1,]    5    7
#  [2,]    6    8
# Could have created in two steps:
A <- 1:8; dim(A) <- c(2,2,2)
# A matrix is a special case of array ...
M <- array(1:9, dim=c(3,3)) # a matrix
# Matrices are arrays with two dimensions
```