# SIOB 296 Introduction to Programming with R

*Eric Archer (eric.archer@noaa.gov)*

*Week 2: January 15, 2019*

### indexing review, vectorization, factors, matrices and arrays

**Reading: The Book of R**

Chapter 3 Matrices and Arrays
Chapter 6.2.4 As-Dot Conversion Functions

---

## Vectorization

A key component of R operations is the idea of "vectorization". It is a built-in capability in R that makes looping over vectors faster. The essence of vectorization is that operations between multiple R vectors will recycle elements in the smaller object to the size of the larger object. This is most easily seen in vector algebra.

```
# Add two vectors of equal length
1:5 + 21:25
```

```
[1] 22 24 26 28 30
```

```
# Add two vectors where one is a multiple of the other
1:10 + 1:2
```

```
 [1]  2  4  4  6  6  8  8 10 10 12
```

```
# Add two vectors where one is not the multiple of the other
1:10 + 1:3
```

```
Warning in 1:10 + 1:3: longer object length is not a multiple of shorter
object length
```

```
 [1]  2  4  6  5  7  9  8 10 12 11
```

Here's an example of vectorization with logical indexing.

```
# Select every other element
x <- 1:10
x[c(T, F)]
```

```
[1] 1 3 5 7 9
```

```
# Select every third element
x[c(T, F, F, F)]
```

```
[1] 1 5 9
```

Whenever possible, take advantage of vectorization. It will be faster than doing explicit loops.

---

## Factors

Factors are special vectors where the unique values are stored as numbers and mapped to character levels

```
x <- factor(c("yellow", "blue", "green", "blue", "Blue", "yellow"))
x
```

```
[1] yellow blue   green  blue   Blue   yellow
Levels: blue Blue green yellow
```

```
# Notice that the values are numerics
str(x)
```

```
 Factor w/ 4 levels "blue","Blue",..: 4 1 3 1 2 4
```

```
# ... but the class isn't
is.numeric(x)
```

```
[1] FALSE
```

```
# ... nor is it character
is.character(x)
```

```
[1] FALSE
```

```
# Here's the class
class(x)
```

```
[1] "factor"
```

```
# and the storage mode
mode(x)
```

```
[1] "numeric"
```

The numeric and original character vectors can be obtained by **coercion** using the `as.<class>` set of functions:

```
as.numeric(x)
```

```
[1] 4 1 3 1 2 4
```

```
as.character(x)
```

```
[1] "yellow" "blue"   "green"  "blue"   "Blue"   "yellow"
```

A factor has both `levels` and `labels`. The `levels` are the set of values that might have existed in the original vector and the `labels` are the representations of the `levels`.

```
# The sample function takes a random sample from a vector with or without replacement
x <- sample(x = letters[1:4], size = 10, replace = TRUE)
xf <- factor(x)
xf
```

```
 [1] a d a c c d a d a d
Levels: a c d
```

```
# Here are the levels
levels(xf)
```

```
[1] "a" "c" "d"
```

```r
# We can change the order of the levels (note doesn't change order of values in vector)
xf.lvl <- factor(x, levels = c("c", "b", "d", "a"))
xf.lvl
```

```
 [1] a d a c c d a d a d
Levels: c b d a
```

```r
# Adding a level that doesn't exist has no effect on data, but includes level in list of levels
xf.lvl <- factor(x, levels = c("c", "e", "b", "d", "a"))
xf.lvl
```

```
 [1] a d a c c d a d a d
Levels: c e b d a
```

```r
# Omitting a level causes all values with that level to be NA
xf.lvl <- factor(x, levels = c("b", "d", "a"))
xf.lvl
```

```
 [1] a    d    a    <NA> <NA> d    a    d    a    d
Levels: b d a
```

```r
# Labels will match order of levels
xf.lbl <- factor(x, labels = c("Z", "Y", "X", "W"))
```

```
Error in factor(x, labels = c("Z", "Y", "X", "W")): invalid 'labels'; length 4 should be 1 or 3
```

```r
xf.lbl
```

```
Error in eval(expr, envir, enclos): object 'xf.lbl' not found
```

```r
# But you must have as many labels as levels
xf.lbl <- factor(x, labels = c("Z", "Y", "X"))
```

---

## Matrices

Matrices are always two-dimensional objects having a certain number of rows and columns. They contain only one kind (atomic mode) of data (e.g., numeric, character, logical). They are created by supplying a vector of values to the `matrix()` function and specifying how many rows and/or how many columns to dimension it by.

```r
# Create a matrix
x <- 1:24
mat <- matrix(x, nrow = 4)
mat
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    5    9   13   17   21
[2,]    2    6   10   14   18   22
[3,]    3    7   11   15   19   23
[4,]    4    8   12   16   20   24
```

```r
# How many elements are in the matrix?
length(mat)
```

```
[1] 24
```

```r
#How many rows and columns?
nrow(mat)
```

```
[1] 4
```

```
ncol(mat)
```

```
[1] 6
```

Cells are selected by `[row, column]`

```
mat[2, 3]
```

```
[1] 10
```

Selecting a single row or single column returns a vector

```
mat[3, ]
```

```
[1]  3  7 11 15 19 23
```

```
mat[, 4]
```

```
[1] 13 14 15 16
```

Use `drop = F` to select a single row or column and return a matrix

```
mat[4, , drop = F]
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    4    8   12   16   20   24
```

```
mat[, 2, drop = F]
```

```
     [,1]
[1,]    5
[2,]    6
[3,]    7
[4,]    8
```

Select several rows or columns

```
mat[c(1, 3, 4), ]
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    5    9   13   17   21
[2,]    3    7   11   15   19   23
[3,]    4    8   12   16   20   24
```

```
mat[, 2:5]
```

```
     [,1] [,2] [,3] [,4]
[1,]    5    9   13   17
[2,]    6   10   14   18
[3,]    7   11   15   19
[4,]    8   12   16   20
```

Select rows, exclude columns

```
mat[1:3, -(2:4)]
```

```
     [,1] [,2] [,3]
[1,]    1   17   21
[2,]    2   18   22
[3,]    3   19   23
```

Change a value in the matrix

```
mat[2, 5] <- NA
```

Change an entire column

```
mat[, 3] <- 100:103
```

Adding a column or row

```
mat.plus.col <- cbind(mat, 100:103)
mat.plus.row <- rbind(300:307, mat)
```

```
Warning in rbind(300:307, mat): number of columns of result is not a
multiple of vector length (arg 1)
```

Assign row and column names

```
rownames(mat) <- c("first", "second", "third", "fourth")
colnames(mat) <- letters[1:ncol(mat)]
```

Choose rows and columns by name

```
mat["first", c("e", "c", "d")]
```

```
  e   c   d
 17 100  13
```

Choose columns by logical vectors

```
mat[, c(T, T, F, F, T, F)]
```

```
       a b  e
first  1 5 17
second 2 6 NA
third  3 7 19
fourth 4 8 20
```

Transpose a matrix

```
t(mat)
```

```
  first second third fourth
a     1      2     3      4
b     5      6     7      8
c   100    101   102    103
d    13     14    15     16
e    17     NA    19     20
f    21     22    23     24
```

Add, subtract, multiply, or divide a matrix by a scalar

```
mat * 5
```

```
        a  b   c  d   e   f
first   5 25 500 65  85 105
second 10 30 505 70  NA 110
third  15 35 510 75  95 115
fourth 20 40 515 80 100 120
```

```
mat / 3
```

```
               a        b        c        d        e        f
first  0.3333333 1.666667 33.33333 4.333333 5.666667 7.000000
```

```
second 0.6666667 2.000000 33.66667 4.666667       NA 7.333333
third  1.0000000 2.333333 34.00000 5.000000 6.333333 7.666667
fourth 1.3333333 2.666667 34.33333 5.333333 6.666667 8.000000
```

```
mat ^ 2
```

```
        a  b     c   d   e   f
first   1 25 10000 169 289 441
second  4 36 10201 196  NA 484
third   9 49 10404 225 361 529
fourth 16 64 10609 256 400 576
```

Add a column

```
mat + 1000:1003
```

```
          a    b    c    d    e    f
first  1001 1005 1100 1013 1017 1021
second 1003 1007 1102 1015   NA 1023
third  1005 1009 1104 1017 1021 1025
fourth 1007 1011 1106 1019 1023 1027
```

Row and column sums or means

```
rowSums(mat)
```

```
 first second  third fourth
   157     NA    169    175
```

```
colMeans(mat)
```

```
    a     b     c     d     e     f
  2.5   6.5 101.5  14.5    NA  22.5
```

---

## Arrays

Arrays are multi-dimensional objects that also contain only a single atomic mode of data. They are indexed the same way as matrices, but created by specifying the number of dimensions.

```
# 1 dimensional array (= vector)
arr.vec <- array(x)
arr.vec
```

```
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
[24] 24
```

```
# 2 dimensional array (= matrix)
arr.mat <- array(x, dim = c(3, 8))
arr.mat
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    1    4    7   10   13   16   19   22
[2,]    2    5    8   11   14   17   20   23
[3,]    3    6    9   12   15   18   21   24
```

```
# 3 dimensional array
arr.3d <- array(x, dim = c(3, 4, 2))
arr.3d
```

```
, , 1

     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12

, , 2

     [,1] [,2] [,3] [,4]
[1,]   13   16   19   22
[2,]   14   17   20   23
[3,]   15   18   21   24
```

The number of dimensions of an object can be obtained with `dim()`.

```
dim(arr.mat)
```

```
[1] 3 8
```

```
dim(arr.3d)
```

```
[1] 3 4 2
```

An array or matrix can be redimensioned as well.

```
dim(arr.mat) <- c(2, 4, 3)
arr.mat
```

```
, , 1

     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8

, , 2

     [,1] [,2] [,3] [,4]
[1,]    9   11   13   15
[2,]   10   12   14   16

, , 3

     [,1] [,2] [,3] [,4]
[1,]   17   19   21   23
[2,]   18   20   22   24
```