Example Rmarkdown Notebook

Adam Lauretig

September 19, 2018

Matrix Algebra and Functions

There are five basic data structures in R: vectors, matrices, arrays, lists, and data.frames. We'll be going through each of these here, but if you want an in depth exploration of these I'd recommend Norman Matloff's *The Art of R Programming: A Tour of Statistical Software Design*.

Matrix basics

Up to this point, we've primarily *talked* about vectors. We've encountered other data types, but haven't used them. Vectors have length, but no width (they can only represent one variable at a time). Matrices are just collections of vectors (exactly like you learned in math camp). We can combine them by column using `cbind`, or by row, using `rbind`. We then access elements of matrix by `matrix[row, column]`.

```
vap <- voting.age.population <- c(3481823, 496387, 4582842, 2120139,26955438,3617942,26731

total.votes <- tv <- c(NA, 238307, 1553032, 780409,8899059,1586105, 1162391,258053, 122356,

m1 <- cbind(vap, tv) # Combined by column
m2 <- rbind(vap, tv) # combined by row
m2[1,2] # first row, second column
```

```
##    vap
## 496387
```

```
m1[,1] # the ith colum
```

```
##  [1]  3481823   496387  4582842  2120139 26955438  3617942  2673154
##  [8]   652189   472143 14085749  6915512   995937  1073799  9600372
## [15]  4732010  2265860  2068253  3213141  3188765  1033632  4242214
## [22]  4997677  7620982  3908159  2139918  4426278   731365  1321923
## [29]  1870315  1012033  6598368  1452962 14838076  6752018   494923
## [36]  8697456  2697855  2850525  9612380   824854  3303593   594599
## [43]  4636679 17038979  1797941   487900  5841335  4876661  1421717
## [50]  4257230   392344
```

```
m1[1:5,1:2] # a submatrix
```

```
##           vap      tv
## [1,]  3481823      NA
## [2,]   496387  238307
## [3,]  4582842 1553032
## [4,]  2120139  780409
## [5,] 26955438 8899059
```

```
m2[1,1:10]
```

```
##  [1]  3481823   496387  4582842  2120139 26955438  3617942  2673154
##  [8]   652189   472143 14085749
```

```
m2[1:2, 1:10]
```

```
##         [,1]   [,2]    [,3]    [,4]     [,5]    [,6]    [,7]   [,8]   [,9]
## vap 3481823 496387 4582842 2120139 26955438 3617942 2673154 652189 472143
## tv       NA 238307 1553032  780409  8899059 1586105 1162391 258053 122356
##        [,10]
## vap 14085749
## tv   4884544
```

```
m2[, 1:10] # same as previous line since there are only two rows.
```

```
##         [,1]   [,2]    [,3]    [,4]     [,5]    [,6]    [,7]   [,8]   [,9]
## vap 3481823 496387 4582842 2120139 26955438 3617942 2673154 652189 472143
## tv       NA 238307 1553032  780409  8899059 1586105 1162391 258053 122356
##        [,10]
## vap 14085749
## tv   4884544
```

```
class(m2)
```

```
## [1] "matrix"
```

However, we can also create matrices directly, we don't need to create vectors first:

```
#Another way to specify a matrix
matrix(1:10, nrow = 5)
```

```
##      [,1] [,2]
## [1,]    1    6
## [2,]    2    7
## [3,]    3    8
## [4,]    4    9
## [5,]    5   10
```

```
matrix(1:10, ncol = 2) #the same
```

```
##      [,1] [,2]
## [1,]    1    6
## [2,]    2    7
## [3,]    3    8
## [4,]    4    9
## [5,]    5   10
```

```
matrix(1:10, nrow = 5, ncol = 2) # the same
```

```
##      [,1] [,2]
## [1,]    1    6
## [2,]    2    7
## [3,]    3    8
## [4,]    4    9
## [5,]    5   10
```

```
matrix(1:10, nrow = 5, byrow = TRUE) ## not the same
```

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

```
## [5,]    9   10
```

By default, R will fill each column of a matrix, and then move to the next one. If you specify `byrow = TRUE`, however, R will fill each row, and then move onto the next one.

Arrays and attributes

Arrays are a more general way to store data. Where a matrix can only have 2 dimensions (rows and columns), arrays can have an arbitrary number of dimensions, but this *will* increase the amount of memory they consume.

Let's examine a cube of dimensions $3 \times 4 \times 2$. One way of thinking of this is two 3×4 matrices stacked on top of each other:

```
a <- array(1:24, dim = c(3, 4, 2))
a
```

```
## , , 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
```

Since this array has three dimensions, there are now three indices we can use to access the array:

```
a[, , 1]
```

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

```
a[, 1, ]
```

```
##      [,1] [,2]
## [1,]    1   13
## [2,]    2   14
## [3,]    3   15
```

```
a[1, , ]
```

```
##      [,1] [,2]
## [1,]    1   13
## [2,]    4   16
## [3,]    7   19
## [4,]   10   22
```

```
a[1, 1, ]
```

```
## [1]  1 13
```

```
a[, 1, 1]
```

```
## [1] 1 2 3
```

```
a[1, 1, 1]
```

```
## [1] 1
```

Notice that the 'dim' is asssigned. This is an "attribute" of the array; attributes are some piece of data associated with the structure that isn't the data itself, and are used to make working with these data easier.

```
dim(a)
```

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

```
attributes(a)
```

```
## $dim
## [1] 3 4 2
```

```
str(a)
```

```
##  int [1:3, 1:4, 1:2] 1 2 3 4 5 6 7 8 9 10 ...
```

Matrices also have this attribute (`dim`), and also have and attribute `dimnames()`, which are strings (technically lists of strings, but we'll get to that in a minute), which allow you to label your observations.

```
dim(m1) # number of rows, number of columns
```

```
## [1] 51  2
```

```
attributes(m1) # there is another attribute here -- the columns have names
```

```
## $dim
## [1] 51  2
##
## $dimnames
## $dimnames[[1]]
## NULL
##
## $dimnames[[2]]
## [1] "vap" "tv"
```

```
dimnames(m1) # we can either assign or get the dimnames attribute
```

```
## [[1]]
## NULL
##
## [[2]]
## [1] "vap" "tv"
```

```
# The first part is the rownames (which we didn't assign)
dimnames(m2) # here the columns have no names
```

```
## [[1]]
## [1] "vap" "tv"
##
## [[2]]
## NULL
```

```
dimnames(m1)[[2]][1]<-"Dracula"
head(m1) # We have re-named the first column to have the name "Dracula"
```

```
##       Dracula      tv
## [1,]  3481823      NA
## [2,]   496387  238307
## [3,]  4582842 1553032
## [4,]  2120139  780409
## [5,] 26955438 8899059
## [6,]  3617942 1586105
```

```
dimnames(m1)[[2]][1]<-"vap" # all of this bracketing is because this is a list ... what's a
head(m1)
```

```
##           vap      tv
## [1,]  3481823      NA
## [2,]   496387  238307
## [3,]  4582842 1553032
## [4,]  2120139  780409
## [5,] 26955438 8899059
## [6,]  3617942 1586105
```

R is flexible, and there are multiple ways to access dimnames:

```
# Another way to do this
colnames(m1)
```

```
## [1] "vap" "tv"
```

```
# How would we rename the first column?
colnames(m2)
```

```
## NULL
```

```
rownames(m1)
```

```
## NULL
```

```
rownames(m2)
```

```
## [1] "vap" "tv"
```

Lists

One downside to matrices and vectors is that every element in them must be the same type (all numerics, or all intergers, or all character vectors). Lists offer a way around this restriction, they can combine multiple data types. Lists are a very flexible way to store data, and are maybe the most common data structure you'll encounter: many functions produce lists.

```
list.a <- list(m1, vap, 3) # m1 is a matrix, vap is a vector, 3 is an integer
list.a
```

```
## [[1]]
##           vap      tv
##  [1,]  3481823      NA
##  [2,]   496387  238307
##  [3,]  4582842 1553032
##  [4,]  2120139  780409
##  [5,] 26955438 8899059
##  [6,]  3617942 1586105
##  [7,]  2673154 1162391
##  [8,]   652189  258053
```

```
##  [9,]   472143  122356
## [10,] 14085749 4884544
## [11,]  6915512 2143845
## [12,]   995937  348988
## [13,]  1073799  458927
## [14,]  9600372 3586292
## [15,]  4732010 1719351
## [16,]  2265860 1071509
## [17,]  2068253  864083
## [18,]  3213141 1370062
## [19,]  3188765  954896
## [20,]  1033632      NA
## [21,]  4242214 1809237
## [22,]  4997677 2243835
## [23,]  7620982 3852008
## [24,]  3908159 2217552
## [25,]  2139918      NA
## [26,]  4426278 2178278
## [27,]   731365  411061
## [28,]  1321923  610499
## [29,]  1870315  586274
## [30,]  1012033  418550
## [31,]  6598368 2315643
## [32,]  1452962  568597
## [33,] 14838076 4703830
## [34,]  6752018 2036451
## [35,]   494923  220479
## [36,]  8697456 4184072
## [37,]  2697855      NA
## [38,]  2850525 1399650
## [39,]  9612380      NA
## [40,]   824854  392882
## [41,]  3303593 1117311
## [42,]   594599  341105
## [43,]  4636679 1868363
## [44,] 17038979      NA
## [45,]  1797941  582561
## [46,]   487900  263025
## [47,]  5841335 2398589
## [48,]  4876661 2085074
## [49,]  1421717  473014
## [50,]  4257230 2183155
## [51,]   392344  196217
##
## [[2]]
##  [1]  3481823   496387  4582842  2120139 26955438  3617942  2673154
##  [8]   652189   472143 14085749  6915512   995937  1073799  9600372
## [15]  4732010  2265860  2068253  3213141  3188765  1033632  4242214
## [22]  4997677  7620982  3908159  2139918  4426278   731365  1321923
## [29]  1870315  1012033  6598368  1452962 14838076  6752018   494923
## [36]  8697456  2697855  2850525  9612380   824854  3303593   594599
## [43]  4636679 17038979  1797941   487900  5841335  4876661  1421717
## [50]  4257230   392344
##
```

```
## [[3]]
## [1] 3
```

We can make all sorts of lists, and can even create lists containing other lists!

```
vector1 <- c(1,2,3)
gospels <- c("matthew","mark","luke", "john")
my.matrix <- matrix(c(1:20), nrow=4)
my.data <- data.frame(cbind(vap, tv))
my.crazy.list <- list(vector1, gospels, my.matrix, TRUE, list.a)
my.crazy.list # we can combine anything we want -- we can even include other lists in our
```

```
## [[1]]
## [1] 1 2 3
##
## [[2]]
## [1] "matthew" "mark"    "luke"    "john"
##
## [[3]]
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    5    9   13   17
## [2,]    2    6   10   14   18
## [3,]    3    7   11   15   19
## [4,]    4    8   12   16   20
##
## [[4]]
## [1] TRUE
##
## [[5]]
## [[5]][[1]]
##             vap      tv
##  [1,]  3481823      NA
##  [2,]   496387  238307
##  [3,]  4582842 1553032
##  [4,]  2120139  780409
##  [5,] 26955438 8899059
##  [6,]  3617942 1586105
##  [7,]  2673154 1162391
##  [8,]   652189  258053
##  [9,]   472143  122356
## [10,] 14085749 4884544
## [11,]  6915512 2143845
## [12,]   995937  348988
## [13,]  1073799  458927
## [14,]  9600372 3586292
## [15,]  4732010 1719351
## [16,]  2265860 1071509
## [17,]  2068253  864083
## [18,]  3213141 1370062
## [19,]  3188765  954896
## [20,]  1033632      NA
## [21,]  4242214 1809237
## [22,]  4997677 2243835
## [23,]  7620982 3852008
## [24,]  3908159 2217552
```

```
## [25,]  2139918      NA
## [26,]  4426278 2178278
## [27,]   731365  411061
## [28,]  1321923  610499
## [29,]  1870315  586274
## [30,]  1012033  418550
## [31,]  6598368 2315643
## [32,]  1452962  568597
## [33,] 14838076 4703830
## [34,]  6752018 2036451
## [35,]   494923  220479
## [36,]  8697456 4184072
## [37,]  2697855      NA
## [38,]  2850525 1399650
## [39,]  9612380      NA
## [40,]   824854  392882
## [41,]  3303593 1117311
## [42,]   594599  341105
## [43,]  4636679 1868363
## [44,] 17038979      NA
## [45,]  1797941  582561
## [46,]   487900  263025
## [47,]  5841335 2398589
## [48,]  4876661 2085074
## [49,]  1421717  473014
## [50,]  4257230 2183155
## [51,]   392344  196217
##
## [[5]][[2]]
##  [1]  3481823   496387  4582842  2120139 26955438  3617942  2673154
##  [8]   652189   472143 14085749  6915512   995937  1073799  9600372
## [15]  4732010  2265860  2068253  3213141  3188765  1033632  4242214
## [22]  4997677  7620982  3908159  2139918  4426278   731365  1321923
## [29]  1870315  1012033  6598368  1452962 14838076  6752018   494923
## [36]  8697456  2697855  2850525  9612380   824854  3303593   594599
## [43]  4636679 17038979  1797941   487900  5841335  4876661  1421717
## [50]  4257230   392344
##
## [[5]][[3]]
## [1] 3
```

What if we want to access the attributes of our list?

```
str(my.crazy.list) # the str() function is useful for looking at the basic components
```

```
## List of 5
##  $ : num [1:3] 1 2 3
##  $ : chr [1:4] "matthew" "mark" "luke" "john"
##  $ : int [1:4, 1:5] 1 2 3 4 5 6 7 8 9 10 ...
##  $ : logi TRUE
##  $ :List of 3
##   ..$ : num [1:51, 1:2] 3481823 496387 4582842 2120139 26955438 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : NULL
##   .. .. ..$ : chr [1:2] "vap" "tv"
```

```
##   ..$ : num [1:51] 3481823 496387 4582842 2120139 26955438 ...
##   ..$ : num 3
```

```
# of any complicated object like this
#str() will work with most types of objects

attributes(my.crazy.list) # lists has attributes, but we haven't set them
```

```
## NULL
```

```
length(my.crazy.list) # this reports the number of major sub-elements in the list
```

```
## [1] 5
```

```
dim(my.crazy.list) # this won't work for complicated lists
```

```
## NULL
```

```
names(my.crazy.list) <- c("one", "two", "three", "four", "five")
str(my.crazy.list) # now each part of the list has a name attribute
```

```
## List of 5
##  $ one  : num [1:3] 1 2 3
##  $ two  : chr [1:4] "matthew" "mark" "luke" "john"
##  $ three: int [1:4, 1:5] 1 2 3 4 5 6 7 8 9 10 ...
##  $ four : logi TRUE
##  $ five :List of 3
##   ..$ : num [1:51, 1:2] 3481823 496387 4582842 2120139 26955438 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : NULL
##   .. .. ..$ : chr [1:2] "vap" "tv"
##   ..$ : num [1:51] 3481823 496387 4582842 2120139 26955438 ...
##   ..$ : num 3
```

```
my.crazy.list
```

```
## $one
## [1] 1 2 3
##
## $two
## [1] "matthew" "mark"    "luke"    "john"
##
## $three
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    5    9   13   17
## [2,]    2    6   10   14   18
## [3,]    3    7   11   15   19
## [4,]    4    8   12   16   20
##
## $four
## [1] TRUE
##
## $five
## $five[[1]]
##            vap      tv
##  [1,]  3481823      NA
##  [2,]   496387  238307
##  [3,]  4582842 1553032
```

```
##  [4,]  2120139  780409
##  [5,] 26955438 8899059
##  [6,]  3617942 1586105
##  [7,]  2673154 1162391
##  [8,]   652189  258053
##  [9,]   472143  122356
## [10,] 14085749 4884544
## [11,]  6915512 2143845
## [12,]   995937  348988
## [13,]  1073799  458927
## [14,]  9600372 3586292
## [15,]  4732010 1719351
## [16,]  2265860 1071509
## [17,]  2068253  864083
## [18,]  3213141 1370062
## [19,]  3188765  954896
## [20,]  1033632      NA
## [21,]  4242214 1809237
## [22,]  4997677 2243835
## [23,]  7620982 3852008
## [24,]  3908159 2217552
## [25,]  2139918      NA
## [26,]  4426278 2178278
## [27,]   731365  411061
## [28,]  1321923  610499
## [29,]  1870315  586274
## [30,]  1012033  418550
## [31,]  6598368 2315643
## [32,]  1452962  568597
## [33,] 14838076 4703830
## [34,]  6752018 2036451
## [35,]   494923  220479
## [36,]  8697456 4184072
## [37,]  2697855      NA
## [38,]  2850525 1399650
## [39,]  9612380      NA
## [40,]   824854  392882
## [41,]  3303593 1117311
## [42,]   594599  341105
## [43,]  4636679 1868363
## [44,] 17038979      NA
## [45,]  1797941  582561
## [46,]   487900  263025
## [47,]  5841335 2398589
## [48,]  4876661 2085074
## [49,]  1421717  473014
## [50,]  4257230 2183155
## [51,]   392344  196217
##
## $five[[2]]
##  [1]  3481823   496387  4582842  2120139 26955438  3617942  2673154
##  [8]   652189   472143 14085749  6915512   995937  1073799  9600372
## [15]  4732010  2265860  2068253  3213141  3188765  1033632  4242214
## [22]  4997677  7620982  3908159  2139918  4426278   731365  1321923
```

```
## [29]  1870315  1012033  6598368  1452962 14838076  6752018   494923
## [36]  8697456  2697855  2850525  9612380   824854  3303593   594599
## [43]  4636679 17038979  1797941   487900  5841335  4876661  1421717
## [50]  4257230   392344
##
## $five[[3]]
## [1] 3
```

But this can be quite convoluted. Instead, when we create our list, we can give each element a name:

```
my.crazy.list <- list(one=vector1,two=gospels, three=my.matrix, four=TRUE, five=list.a)
str(my.crazy.list)
```

```
## List of 5
##  $ one  : num [1:3] 1 2 3
##  $ two  : chr [1:4] "matthew" "mark" "luke" "john"
##  $ three: int [1:4, 1:5] 1 2 3 4 5 6 7 8 9 10 ...
##  $ four : logi TRUE
##  $ five :List of 3
##   ..$ : num [1:51, 1:2] 3481823 496387 4582842 2120139 26955438 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : NULL
##   .. .. ..$ : chr [1:2] "vap" "tv"
##   ..$ : num [1:51] 3481823 496387 4582842 2120139 26955438 ...
##   ..$ : num 3
```

```
names(my.crazy.list)
```

```
## [1] "one"   "two"   "three" "four"  "five"
```

Manipulating lists is similar to other manipulations in R, the new one is using double brackets `[[]]` to access an element of a list.

```
# there are several ways to access/add to/subtract from a list
my.crazy.list[[1]]
```

```
## [1] 1 2 3
```

```
my.crazy.list$one
```

```
## [1] 1 2 3
```

```
my.crazy.list[1]
```

```
## $one
## [1] 1 2 3
```

```
my.crazy.list["one"]
```

```
## $one
## [1] 1 2 3
```

```
my.crazy.list$dracula <- "dracula"
my.crazy.list # now we have added another element
```

```
## $one
## [1] 1 2 3
##
## $two
## [1] "matthew" "mark"    "luke"    "john"
##
```

```
## $three
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    5    9   13   17
## [2,]    2    6   10   14   18
## [3,]    3    7   11   15   19
## [4,]    4    8   12   16   20
##
## $four
## [1] TRUE
##
## $five
## $five[[1]]
##            vap      tv
##  [1,]  3481823      NA
##  [2,]   496387  238307
##  [3,]  4582842 1553032
##  [4,]  2120139  780409
##  [5,] 26955438 8899059
##  [6,]  3617942 1586105
##  [7,]  2673154 1162391
##  [8,]   652189  258053
##  [9,]   472143  122356
## [10,] 14085749 4884544
## [11,]  6915512 2143845
## [12,]   995937  348988
## [13,]  1073799  458927
## [14,]  9600372 3586292
## [15,]  4732010 1719351
## [16,]  2265860 1071509
## [17,]  2068253  864083
## [18,]  3213141 1370062
## [19,]  3188765  954896
## [20,]  1033632      NA
## [21,]  4242214 1809237
## [22,]  4997677 2243835
## [23,]  7620982 3852008
## [24,]  3908159 2217552
## [25,]  2139918      NA
## [26,]  4426278 2178278
## [27,]   731365  411061
## [28,]  1321923  610499
## [29,]  1870315  586274
## [30,]  1012033  418550
## [31,]  6598368 2315643
## [32,]  1452962  568597
## [33,] 14838076 4703830
## [34,]  6752018 2036451
## [35,]   494923  220479
## [36,]  8697456 4184072
## [37,]  2697855      NA
## [38,]  2850525 1399650
## [39,]  9612380      NA
## [40,]   824854  392882
## [41,]  3303593 1117311
```

```
## [42,]   594599  341105
## [43,]  4636679 1868363
## [44,] 17038979      NA
## [45,]  1797941  582561
## [46,]   487900  263025
## [47,]  5841335 2398589
## [48,]  4876661 2085074
## [49,]  1421717  473014
## [50,]  4257230 2183155
## [51,]   392344  196217
##
## $five[[2]]
##  [1]  3481823   496387  4582842  2120139 26955438  3617942  2673154
##  [8]   652189   472143 14085749  6915512   995937  1073799  9600372
## [15]  4732010  2265860  2068253  3213141  3188765  1033632  4242214
## [22]  4997677  7620982  3908159  2139918  4426278   731365  1321923
## [29]  1870315  1012033  6598368  1452962 14838076  6752018   494923
## [36]  8697456  2697855  2850525  9612380   824854  3303593   594599
## [43]  4636679 17038979  1797941   487900  5841335  4876661  1421717
## [50]  4257230   392344
##
## $five[[3]]
## [1] 3
##
##
## $dracula
## [1] "dracula"
```

```
# We can repeat this accessing method
my.crazy.list[[3]][1,] # first row of my.matrix
```

```
## [1]  1  5  9 13 17
```

```
my.matrix[1,]  #the same
```

```
## [1]  1  5  9 13 17
```

However, you cannot do math on lists directly (note that this is set to `eval = FALSE`, since if we ran it, it throws an error and the document doesn't compile):

```
my.crazy.list +2 # not so much
my.crazy.list[[3]] + 2
```