Lab 8

Maria Barouti

2/25/2020

Bulding matrices

```
m <- matrix(c(1, 0, 0, 1), nrow = 2, ncol = 2, byrow = TRUE)
m
```

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
```

You can test whether an item you have been given is a matrix using `is.matrix` and you can convert appropriate objects to matrices using `as.matrix`

```
m <- matrix(c(1, 0, 0, 1), nrow = 2, ncol = 2, byrow = TRUE)

is.matrix(m)
```

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

```
as.matrix(data.frame(x = c(1, 0), y = c(0, 1)))
```

```
##      x y
## [1,] 1 0
## [2,] 0 1
```

Diagonal matrices

```
n <- 10
m <- diag(nrow = n, ncol = n)
m
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]    1    0    0    0    0    0    0    0    0     0
##  [2,]    0    1    0    0    0    0    0    0    0     0
##  [3,]    0    0    1    0    0    0    0    0    0     0
##  [4,]    0    0    0    1    0    0    0    0    0     0
##  [5,]    0    0    0    0    1    0    0    0    0     0
##  [6,]    0    0    0    0    0    1    0    0    0     0
##  [7,]    0    0    0    0    0    0    1    0    0     0
##  [8,]    0    0    0    0    0    0    0    1    0     0
##  [9,]    0    0    0    0    0    0    0    0    1     0
## [10,]    0    0    0    0    0    0    0    0    0     1
```

I will be exploiting a more interesting use of `diag` in my next post, so let's see how you can build matrices other than the identity with `diag` by specifying a vector of entries along the diagonal.

```
m <- diag(c(2, 1), nrow = 2, ncol = 2)
m
```

```
##      [,1] [,2]
## [1,]    2    0
## [2,]    0    1
```

Matrix algebra: Addition, scalar Multiplication, matrix Multiplication

```r
m <- matrix(c(0, 2, 1, 0), nrow = 2, ncol = 2, byrow = TRUE)
m
```

```
##      [,1] [,2]
## [1,]    0    2
## [2,]    1    0
```

```r
# Addition
m + m
```

```
##      [,1] [,2]
## [1,]    0    4
## [2,]    2    0
```

```r
# Scalar multiplication
2 * m
```

```
##      [,1] [,2]
## [1,]    0    4
## [2,]    2    0
```

```r
# Matrix multiplication
m %*% m
```

```
##      [,1] [,2]
## [1,]    2    0
## [2,]    0    2
```

Note: Be careful with the `*` operator: it does not perform matrix multiplication, but rather an entry-wise multiplication:

```r
m * m
```

```
##      [,1] [,2]
## [1,]    0    4
## [2,]    1    0
```

Matrix transposes and inverses

To get the transpose of a matrix, you simply call the `t` function:

```r
t(m)
```

```
##      [,1] [,2]
## [1,]    0    1
## [2,]    2    0
```

In contrast, inversion is a little more complex, partly because the function you want to use has a non-obvious name: `solve`.

```r
solve(m)
```

```
##      [,1] [,2]
## [1,]  0.0    1
## [2,]  0.5    0
```

Now, you probably know this already, but the definition of a matrix's inverse is that the product of the matrix and its inverse is the identity matrix, if the inverse exists. I always find this a good way to make sure that I am correctly computing the inverse of a matrix:

```r
solve(m) %*% m == diag(nrow = nrow(m), ncol = ncol(m))
```

```
##      [,1] [,2]
## [1,] TRUE TRUE
## [2,] TRUE TRUE
```

Eigenvalues and eigenvectors

```r
m <- diag(nrow = 2, ncol = 2)

eigen(m)
```

```
## eigen() decomposition
## $values
## [1] 1 1
##
## $vectors
##      [,1] [,2]
## [1,]    0   -1
## [2,]    1    0
```

Matrix metadata

```r
m <- diag(nrow = 2, ncol = 2)
dim(m)
```

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

```r
nrow(m)
```

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

```r
ncol(m)
```

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