Matrix Algebra

Gaston Sanchez

November 7, 2016

Basic Vector and Matrix manipulations in R

If you have no or minimal experience in R working with matrices, please go over this section. If you are already familiar with this material, skip to the next section.

Consider the following vector `x`:

```
x <- 1:9
```

- Use the vector `x` as input of the function `matrix()` to create the following matrix:

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

- Using `x` and `matrix()`, how would you generate a matrix like this:

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

- Use `diag()` to create the following identity matrix $\mathbf{I}_n$ of dimensions $n \times p = (5, 5)$:

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    0    0    0
[2,]    0    1    0    0    0
[3,]    0    0    1    0    0
[4,]    0    0    0    1    0
[5,]    0    0    0    0    1
```

- Consider the following vectors `a1, a2, a3`:

```
a1 <- c(2, 3, 6, 7, 10)
a2 <- c(1.88, 2.05, 1.70, 1.60, 1.78)
a3 <- c(80, 90, 70, 50, 75)
```

- Column-bind the vectors `a1, a2, a3` to form this matrix:

```
  a1   a2 a3
1  2 1.88 80
2  3 2.05 90
3  6 1.70 70
4  7 1.60 50
5 10 1.78 75
```

- Now row-bind the vectors `a1, a2, a3` to form this matrix:

```
       1     2    3    4     5
a1  2.00  3.00  6.0  7.0 10.00
a2  1.88  2.05  1.7  1.6  1.78
a3 80.00 90.00 70.0 50.0 75.00
```

- Using matrix functions like transpose `t()` and product `%*%`, write a function `vnorm()` that computes the Euclidean norm (i.e. length) of a vector: $\|\mathbf{x}\| = \sqrt{\mathbf{x}^\mathsf{T}\mathbf{x}}$.

```
# test vnorm() with 1:5
v <- 1:5
vnorm(v)
```

- Given the vector `v <- 1:5`, use `vnorm()` to create a vector `u` such that `u` is of unit norm: $\|\mathbf{u}\| = 1$.
- Write a function `is_square()` to check whether the provided matrix is a square matrix.
- Write a function `mtrace()` to compute the trace of a square matrix. Use your `is_square()` function to make sure that the input is a square matrix. The trace is defined as the sum of the elements on the diagonal: $tr(\mathbf{A}) = \sum_{i=1}^{n} a_{ii}$
- Given two square matrices $\mathbf{A}$ and $\mathbf{B}$, verify that your function `tr()` is a linear mapping:
  - $tr(\mathbf{A} + \mathbf{B}) = tr(\mathbf{A}) + tr(\mathbf{A})$
  - $tr(c\mathbf{A}) = c \times tr(\mathbf{A})$, where c is a scalar
- Trace of products: Given two matrices $\mathbf{X}$ and $\mathbf{Y}$ of adequate size, verify that:

$$tr(\mathbf{X}^\mathsf{T}\mathbf{Y}) = tr(\mathbf{X}\mathbf{Y}^\mathsf{T}) = tr(\mathbf{Y}^\mathsf{T}\mathbf{X}) = tr(\mathbf{Y}\mathbf{X}^\mathsf{T})$$

- Cross-products: Use `proc.time()` or `system.time()` to compare the `t(X) %*% X` against `crossprod(X)`
- Cross-products: Use `proc.time()` or `system.time()` to compare the `X %*% t(X)` against `tcrossprod(X)`

Transformation and Scaling Operations

In this section you will be using the built in data.frame `mtcars` to practice transforming and scaling operations:

```
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

- Create a matrix `M` with variables `mpg`, `disp`, `hp`, `drat`, and `wt`.
- Use `apply()` to compute the vector containing the means of the columns in `M`
- Compute a matrix `Mc` of mean-centered data applying the function `scale()` on `M` (do NOT `scale = TRUE`).
- Confirm that variables in `Mc` are mean-centered by calculating the vector of column-means
- Use the function `sweep()` to mean-center `M` by *sweeping out* the vector of column means. Compare this result with `Mc` (you should get the same values).
- Compute a vector of column maxima.
- Use `sweep()` to scale the columns by dividing by the column maxima.
- Compute a matrix in which all columns are scaled to have minimum $= 0$ and maximum $= 1$
- Write a function `dummify()` that takes a factor or a character vector, and which returns a matrix with dummy indicators. Assuming that the factor has k categories (i.e. k `levels`), include an argument `all` that lets you specify whether to return k binary indicators, or $k - 1$ indicators. You should be able to call `dummify()` like this:

```
cyl <- factor(mtcars$cyl)
# all categories
CYL1 <- dummify(cyl, all = TRUE)
# minus one category
CYL2 <- dummify(cyl, all = FALSE)
```

- Write a function `crosstable()` that takes two factors, and which returns a cross-table between those factors. To create this function you should use your function `dummify()` to compute two dummy matrices, and then multiple them.

```
cyl <- factor(mtcars$cyl)
gear <- factor(mtcars$gear)
xtb <- crosstable(cyl, gear)
```

You should get a table like this:

```
   3 4 5
4  1 8 2
6  2 4 1
8 12 0 2
```

Solutions

```r
M1 <- matrix(x, nrow = 3, ncol = 3)
M1
```

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

```r
M2 <- matrix(x, nrow = 3, ncol = 3, byrow = TRUE)
M2
```

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

```r
diag(1, nrow = 5)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    0    0    0    0
## [2,]    0    1    0    0    0
## [3,]    0    0    1    0    0
## [4,]    0    0    0    1    0
## [5,]    0    0    0    0    1
```

```r
M3 <- cbind(a1, a2, a3)
rownames(M3) <- 1:nrow(M3)
M3
```

```
##   a1   a2 a3
## 1  2 1.88 80
## 2  3 2.05 90
## 3  6 1.70 70
## 4  7 1.60 50
## 5 10 1.78 75
```

```r
M4 <- rbind(a1, a2, a3)
colnames(M4) <- 1:ncol(M4)
M4
```

```
##        1     2    3    4     5
## a1  2.00  3.00  6.0  7.0 10.00
## a2  1.88  2.05  1.7  1.6  1.78
## a3 80.00 90.00 70.0 50.0 75.00
```

```r
# vector norm
vnorm <- function(x) {
  sqrt(t(x) %*% x)
}
```

```r
vnorm(1:5)
```

```
##          [,1]
## [1,] 7.416198
```

```r
# vector of unit norm
v <- 1:5
u <- v / vnorm(v)
vnorm(u)
```

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

```r
# is a matrix square
is_square <- function(m) {
  nrow(m) == ncol(m)
}

is_square(matrix(1:9, 3, 3))
```

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

```r
# function for trace of a matrix
mtrace <- function(m) {
  if (!is_square(m)) {
    stop("\n'mtrace()' requires a square matrix")
  }
  sum(diag(m))
}

mtrace(matrix(1:9, nrow = 3, ncol = 3))
```

```
## [1] 15
```

```r
# trace is a linear mapping
set.seed(123)
A <- matrix(sample(1:9, 9), nrow = 3, ncol = 3)
B <- matrix(sample(1:9, 9), nrow = 3, ncol = 3)

mtrace(A + B)
```

```
## [1] 22
```

```r
mtrace(A) + mtrace(B)
```

```
## [1] 22
```

```r
mtrace(3 * A)
```

```
## [1] 36
```

```r
3 * mtrace(A)
```

```
## [1] 36
```

```r
# trace of a product
set.seed(123)
X <- matrix(sample(1:12, 12), nrow = 4, ncol = 3)
Y <- matrix(sample(1:12, 12), nrow = 4, ncol = 3)

mtrace(t(X) %*% Y)
```

```
## [1] 593
```

```r
mtrace(X %*% t(Y))
```

```
## [1] 593
```

```r
mtrace(t(Y) %*% X)
```

```
## [1] 593
```

```r
mtrace(Y %*% t(X))
```

```
## [1] 593
```

```r
# crossproduct
set.seed(345)
X <- matrix(runif(100000), 1000, 100)
system.time(t(X) %*% X)
```

```
##    user  system elapsed
##   0.007   0.000   0.007
```

```r
system.time(crossprod(X))
```

```
##    user  system elapsed
##   0.005   0.000   0.004
```

```r
# tcrossproduct
set.seed(345)
Y <- matrix(runif(100000), 100, 1000)
system.time(Y %*% t(Y))
```

```
##    user  system elapsed
##   0.007   0.001   0.008
```

```r
system.time(tcrossprod(Y))
```

```
##    user  system elapsed
##   0.004   0.000   0.004
```

```r
M <- as.matrix(mtcars[ ,c('mpg', 'disp', 'hp', 'drat', 'wt')])

# vector of column-means
means <- apply(M, MARGIN = 2, FUN = mean)

# mean-centered Mc with scale()
```

```r
Mc <- scale(M, scale = FALSE)
apply(Mc, MARGIN = 2, FUN = mean)
```

```
##           mpg          disp            hp          drat            wt
##  4.440892e-16 -1.199041e-14  0.000000e+00 -1.526557e-16  3.469447e-17
```

```r
# mean-centered with sweep()
Mc2 <- sweep(M, MARGIN = 2, STATS = means, FUN = "-")
apply(Mc2, MARGIN = 2, FUN = mean)
```

```
##           mpg          disp            hp          drat            wt
##  4.440892e-16 -1.199041e-14  0.000000e+00 -1.526557e-16  3.469447e-17
```

```r
# scaling by the maximum
maxs <- apply(M, MARGIN = 2, FUN = max)
Mx <- sweep(M, MARGIN = 2, STATS = maxs, FUN = "/")

# rescaling from 0 to 1
mins <- apply(M, MARGIN = 2, FUN = min)
ranges <- apply(M, MARGIN = 2, FUN = function(x) max(x) - min(x))
M01 <- scale(M, center = mins, scale = ranges)
# test it
apply(M01, MARGIN = 2, FUN = min)
```

```
##  mpg disp   hp drat   wt
##    0    0    0    0    0
```

```r
apply(M01, MARGIN = 2, FUN = max)
```

```
##  mpg disp   hp drat   wt
##    1    1    1    1    1
```

```r
# function dummify
dummify <- function(x, all = FALSE) {
  if (!is.factor(x)) x <- as.factor(x)
  categories <- levels(x)
  num_categories <- length(categories)
  if (!all) {
    num_categories <- length(categories) - 1
  }
  dummies <- matrix(0, nrow = length(x), ncol = num_categories)
  for (k in 1:num_categories) {
    dummies[x == categories[k],k] <- 1
  }
  colnames(dummies) <- categories[1:num_categories]
  dummies
}

# test it
cyl <- factor(mtcars$cyl)
```

```r
CYL1 <- dummify(cyl, all = TRUE)
CYL2 <- dummify(cyl, all = FALSE)
```

```r
# function crosstable
crosstable <- function(x, y) {
  if (!is.factor(x)) x <- factor(x)
  if (!is.factor(y)) y <- factor(y)
  Xdum <- dummify(x, all = TRUE)
  Ydum <- dummify(y, all = TRUE)
  t(Xdum) %*% Ydum
}

cyl <- factor(mtcars$cyl)
gear <- factor(mtcars$gear)
xtb <- crosstable(cyl, gear)
```