R 프로그래밍 기초다지기

3강 - 행렬? 그까이꺼

슬기로운통계생활

Issac Lee

행렬 그까이꺼

행렬이란 무엇일까?

벡터들을 모아놓은 것

- 꼭! 사각형 모양
- `dim()`로 사각형의 크기 잴 수 있음

```
cbind(1:4, 12:15)
```

```
##      [,1] [,2]
## [1,]    1   12
## [2,]    2   13
## [3,]    3   14
## [4,]    4   15
```

```
dim(cbind(1:4, 12:15))
```

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

행렬 선언하기

`matrix()` 함수

- 행과 열 중 하나만 입력해도 자동으로 계산해서 만들어줌
- 위치는 대괄호 안에 순서쌍으로 나타냄
  - 문법: `[row, col]`
  - 1행 2열에 위치한 원소: `[1, 2]`

```
y <- matrix(1:4, nrow = 2)
y
```

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

```
y[1, 2]
```

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

행렬을 채우는 방법 - `byrow` 옵션

- `byrow` 옵션을 통하여 행렬에 숫자를 채우는 방향을 정해줄 수 있음.

- 세로로 채우기

```
matrix(1:6, nrow = 2)
```

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

- 가로로 채우기

```
matrix(1:6, nrow = 2,
             byrow = TRUE)
```

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

행렬의 원소에 접근하기

인덱싱 (Indexing)

```
x <- matrix(1:10 * 2, ncol = 2)
x
```

```
##      [,1] [,2]
## [1,]    2   12
## [2,]    4   14
## [3,]    6   16
## [4,]    8   18
## [5,]   10   20
```

- 전체를 나타내는 빈 칸

```
x[, 2]
```

```
## [1] 12 14 16 18 20
```

- 선택적으로 골라오기

```
x[c(2, 3, 5), 2]
```

```
## [1] 14 16 20
```

필터링

- `TRUE`, `FALSE` 벡터를 사용해서 필터링이 됨

```
x[c(TRUE, TRUE, FALSE, FALSE, TRUE), 1]
```

```
## [1]  2  4 10
```

- 조건문 사용한 필터 가능

```
x[x[, 2] > 15, 1]
```

```
## [1]  6  8 10
```

사진도 행렬이다.

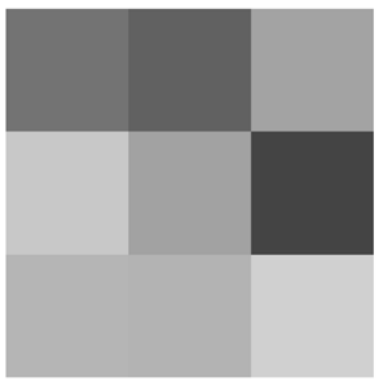
행렬로 사진만들기

- 검정색 (0) 과 흰색 (1) 을 0과 1사이 숫자들과 연결

```
set.seed(2021)
img1 <- matrix(runif(9),
               ncol = 3)
img1
```

```
##           [,1]      [,2]
## [1,] 0.4512674 0.3817443 0.6
## [2,] 0.7837798 0.6363238 0.2
## [3,] 0.7096822 0.7013460 0.8
```

```
plot(raster::as.raster(img1),
     interpolate = FALSE)
```



사진도 행렬이다.

```
data_url <- "https://raw.githubu
download.file(data_url,
              "mat.rds")
img_mat <- readRDS("mat.rds")
file.remove("mat.rds")
```

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

```
dim(img_mat)
```

```
## [1] 88 50
```

```
head(img_mat[1:3, 1:4])
```

```
##      [,1] [,2] [,3] [,4]
## [1,]  132  131  134  132
## [2,]  135  137  137  138
## [3,]  143  142  145  146
```

행렬에서 사진으로

```
max(img_mat)
```

```
## [1] 255
```

```
min(img_mat)
```

```
## [1] 0
```

- 색깔 매칭을 위해 스케일링

```
img_mat <- img_mat/255
```

```
library(raster)
plot(as.raster(img_mat))
```

행렬 클래스

- `class()` 함수를 통해 우리가 만든 행렬이 `matrix`, `array` (행렬의 확장 개념) 클래스라는 것을 확인
- 특정 클래스에는 접근 가능한 속성(`attribute`) 들이 정의되어 있음.

```
class(x)
```

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

```
attributes(x)
```

```
## $dim
## [1] 5 2
```

행렬 뒤집기

Transpose

- `t()` 함수를 사용

```
x
```

```
##      [,1] [,2]
## [1,]    2   12
## [2,]    4   14
## [3,]    6   16
## [4,]    8   18
## [5,]   10   20
```

```
t(x)
```

```
##      [,1] [,2] [,3] [,4] [,5
## [1,]    2    4    6    8    1
## [2,]   12   14   16   18   2
```

행렬의 연산

행렬의 곱셈 `%*%`

- 행렬의 곱셈은 크기가 맞아야지 가능

```
dim(x)
```

```
## [1] 5 2
```

```
dim(y)
```

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

```
x %*% y
```

```
##      [,1] [,2]
## [1,]   26   54
## [2,]   32   68
## [3,]   38   82
## [4,]   44   96
## [5,]   50  110
```

행렬의 연산

원소별 곱셈 (Hadamard product, element-wise product)

```
y
```

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

```
z <- matrix(10:13, ncol = 2)
z
```

```
##      [,1] [,2]
## [1,]   10   12
## [2,]   11   13
```

```
y * z
```

```
##      [,1] [,2]
## [1,]   10   36
## [2,]   22   52
```

행렬의 역행렬

solve() 함수

- `solve()`에 행렬이 입력값으로 들어가면 역행렬이 구해지도록 설계됨.

```
solve(y)
```

```
##      [,1] [,2]
## [1,]   -2  1.5
## [2,]    1 -0.5
```

- 역행렬이 존재하지 않는 경우

```
no_inverse <- matrix(c(1, 2, 1,
solve(no_inverse)
```

```
## Error in solve.default(no_in
```

행렬의 연산과 recyling

행렬 연산에서의 recyling

- 행렬에 벡터를 곱하면, 벡터 길이를 맞춰 계산하듯, 행렬 크기도 맞춰서 계산
- 단, 행렬과 행렬 계산에서는 적용 안됨.

```
y * matrix(c(1, 2), ncol = 1)
```

```
## Error in y * matrix(c(1, 2),
```

```
y
```

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

```
y * c(1, 2)
```

```
##      [,1] [,2]
## [1,]    1    3
## [2,]    4    8
```

차원 축소 효과

가끔 너무 친절해도 불편함.

- 자동 차원 축소

```
y[1, ]
```

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

```
dim(y[1, ])
```

```
## NULL
```

- 차원 축소 기능 끄기

```
y[1, , drop = FALSE]
```

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

```
dim(y[1, , drop = FALSE])
```

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

행렬에 이름 붙이기

```
y
```

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

```
colnames(y)
```

```
## NULL
```

```
colnames(y) <- c("col_1", "col_2")
y
```

```
##      col_1 col_2
## [1,]     1     3
## [2,]     2     4
```

```
rownames(y) <- c("row_1", "row_2")
y
```

```
##       col_1 col_2
## row_1     1     3
## row_2     2     4
```

배열 `array()`

고차원 행렬

- 행렬을 붙여놓을 수 있지 않을까?

```
mat1 <- matrix(1:6, nrow = 2)
mat2 <- matrix(7:12, nrow = 2)
my_array <- array(
  data = c(mat1, mat2),
  dim = c(2, 3, 2)
)
```

```
my_array
```

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

배열 다루기

- 행렬의 필터링 접근 방식이 그대로 적용됨

```
my_array[, , 1]
```

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

```
my_array[, -3, ]
```

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

배열 차원 다루기

배열에서의 transpose

```
my_array
```

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

```
aperm(my_array, c(2,1,3))
```

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

사진은 배열이다.

```
library(png)
img_url <- "https://raw.githubu
download.file(img_url, "jelly.pr
                  mode="wb")
jelly <- readPNG("jelly.png")
file.remove("jelly.png")
```

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

```
dim(jelly)
```

```
## [1] 88 50  4
```

- Red, Green, Blue, Opacity
  - 색깔 속성을 0~255
  - 투명토는 0~1

다음시간

리스트

참고자료 및 사용교재

[1] The art of R programming

- R 공부하시는 분이면 꼭 한번 보셔야하는 책입니다.
- 위 교재의 한글 번역본 빅데이터 분석 도구 R 프로그래밍도 있습니다. 도서 제목 클릭하셔서 구매하시면 저의 사리사욕을 충당하는데 도움이 됩니다.

[2] how to download and display an image from an URL in R?