벡터

Jeon Jong-June

Monday, March 09, 2015

행렬 만들기

행렬 만들기는 matrix함수를 이용한다. nrow는 행의 개수, ncol은 열의 개수다.

```
y <- matrix( c(1, 2, 3, 4), nrow = 2, ncol = 2)
y
```

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

```
y <- matrix( c(1, 2, 3, 4), nrow = 2, ncol = 2, byrow = T)
y
```

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

행렬에서의 원소 골라내기

행렬에서 원소위치는 [행, 열]로 나타낸다. 열 전체, 행 전체를 선택하기 위해서는 열, 행에 해당하는 위치 인덱스를 생략한다.

```
y <- matrix(c(1,3,4,5,1,3,4,1),4,2)
y[1, 1]
```

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

```
y[, 1]
```

```
## [1] 1 3 4 5
```

```
y[1, ]
```

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

```
y[2:3,]
```

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

행렬관련 함수

```
attributes(y)
```

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

```
class(y)
```

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

```
dim(y)
```

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

```
ncol(y)
```

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

```
nrow(y)
```

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

행렬의 연산

- 행렬과 스칼라 곱: A+B
- 행렬의 덧셈: aA
- *행렬의 곱셈:* A%%B
- 역행렬: solve(A)
- 행렬식: det(A)
- 고유값: svd(A)$d

```
A <- matrix(c(1, 0, -1, 2, 1, 0, 1, 1, 3), 3 ,3 )
B <- matrix(rep(c(1,2,3), each = 3), 3, 3)
A+B
```

```
##      [,1] [,2] [,3]
## [1,]    2    4    4
## [2,]    1    3    4
## [3,]    0    2    6
```

```
(2.1)*A
```

```
##      [,1] [,2] [,3]
## [1,]  2.1  4.2  2.1
## [2,]  0.0  2.1  2.1
## [3,] -2.1  0.0  6.3
```

```
A%*%B
```

```
##      [,1] [,2] [,3]
## [1,]    4    8   12
## [2,]    2    4    6
## [3,]    2    4    6
```

```
solve(A)
```

```
##      [,1] [,2] [,3]
## [1,]  1.5   -3  0.5
## [2,] -0.5    2 -0.5
## [3,]  0.5   -1  0.5
```

```
det(A)
```

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

```
svd(A)$d
```

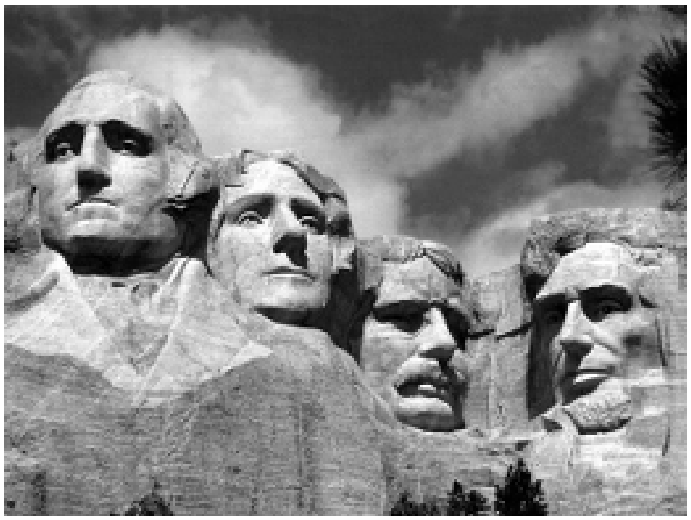
```
## [1] 3.5163959 2.3615577 0.2408428
```

행렬 연산예제

```
library(pixmap)
mtrush1 <-read.pnm("C:\\Users\\uos_stat\\Dropbox\\class\\2015 prog\\lecture note\\Lecture 3\\mtrush1.pgm
mtrush1
```

```
## Pixmap image
##   Type          : pixmapGrey
##   Size          : 194x259
##   Resolution    : 1x1
##   Bounding box  : 0 0 259 194
```

```
plot(mtrush1)
```



```
str(mtrush1)
```

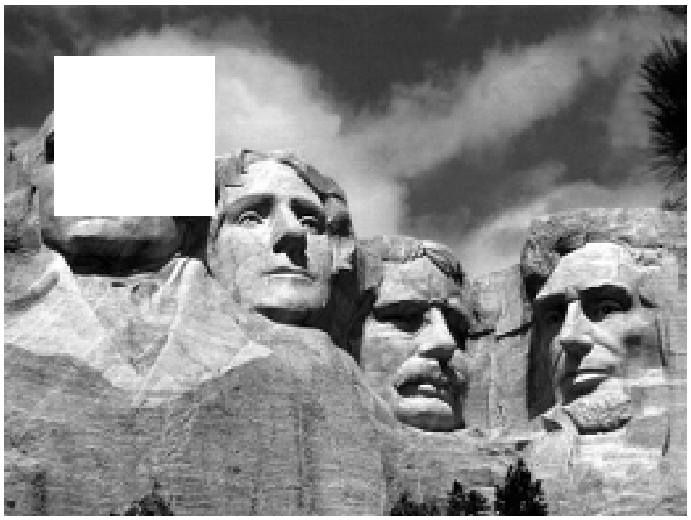
```
## Formal class 'pixmapGrey' [package "pixmap"] with 6 slots
##   ..@ grey    : num [1:194, 1:259] 0.31 0.259 0.231 0.216 0.204 ...
##   ..@ channels: chr "grey"
##   ..@ size    : int [1:2] 194 259
##   ..@ cellres : num [1:2] 1 1
##   ..@ bbox    : num [1:4] 0 0 259 194
##   ..@ bbcent  : logi FALSE
```

locator 함수를 사용해보자

```
locator()
```

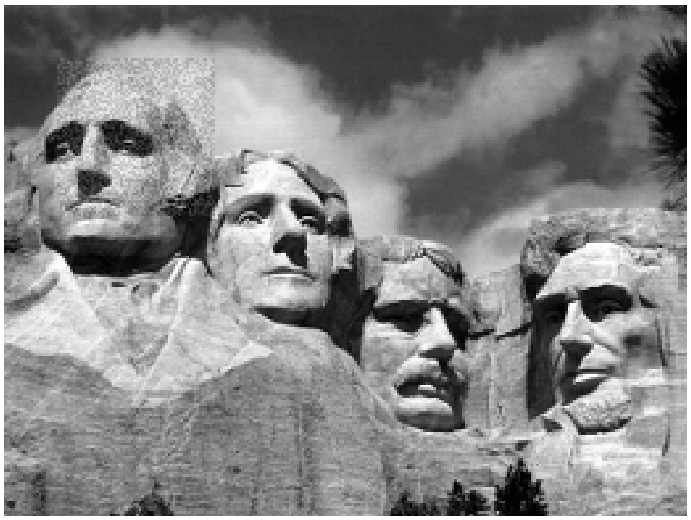
원하는 위치를 클릭한후 ESC를 누르자. 그러면 원하는 위치의 값이 출력된다.

```
x <- mtrush1
x@grey[20:80 , 20:80] <- 1
plot(x)
```

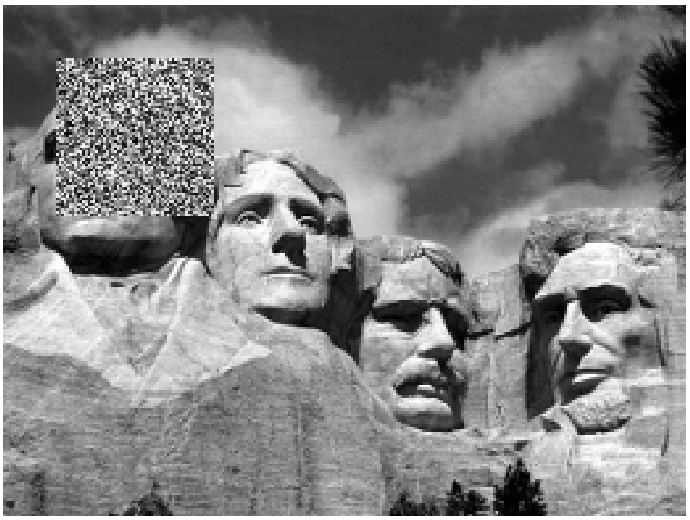


noise 추가

```
x <- mtrush1
y1 <- x@grey[21:80 , 21:80]
t <- 0.8
x@grey[21:80 , 21:80]  <-
    t*y1 + (1-t)*matrix( runif(length(y1)), nrow(y1), ncol(y1))
plot(x)
```



```
x <- mtrush1
y1 <- x@grey[21:80 , 21:80]
t <- 0.1
x@grey[21:80 , 21:80]  <-
    t*y1 + (1-t)*matrix( runif(length(y1)), nrow(y1), ncol(y1))
plot(x)
```



행렬의 필터링

```
x <- matrix(c( 1,2,3,2,3,4),3,2)
x
```

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

```
x[x[, 2]>=3, ]
```

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

행렬의 필터링 역시 x[, 2]>=3 의 논리연산의 결과를 이용함을 확인해보자

```
x[, 2]>=3
```

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

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

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

논리 연산자

```
 z <- 4
 z==4
```

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

```
 z>=4
```

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

```
 z<4
```

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

```
 z!=4
```

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

행과 열 추가 및 제거하기

cbind (column bind) 및 rbind(row bind)함수를 사용하여 행렬의 크기를 늘릴 수 있다.

```
one <- rep(1,4)
z <- matrix( c(1:4, rep(1,4), rep(0,4)), 4, 3)
z
```

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

```
cbind(one, z)
```

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

```
z <- cbind(1, z)
z
```

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

```
z <- rbind(2, z)
z
```

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

행렬에서의 행 및 열의 제거는 - 를 사용하면 된다.

```
z[-2,]
```

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

```
z[,-1]
```

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

```
z[-(1:2),]
```

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

벡터와 행렬

```r
z <- matrix(1:8, 4, 2)
z
```

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

```r
length(z)
```

```
## [1] 8
```

```r
class(z)
```

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

```r
attributes(x)
```

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

의도하지 않은 차원 축소 피하기

```r
z1 <- z[,-1]
class(z1)
```

```
## [1] "integer"
```

```r
z2 <- z[,-1, drop = F]
class(z2)
```

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

벡터에서 행렬로 class 변경

```r
 z1 = as.matrix(z1)
 class(z1)
```

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