

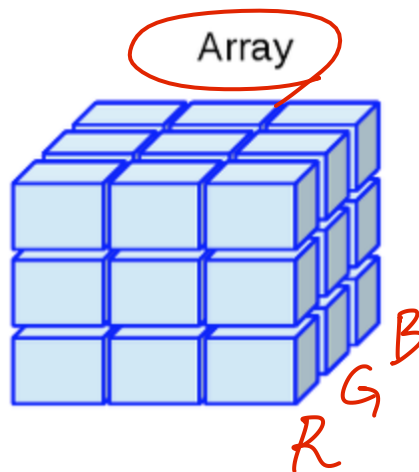
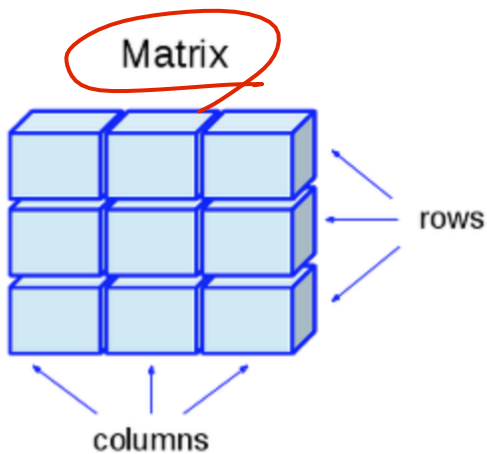
# PCA 실습 2

## 1. 데이터 확인



```
library(jpeg)
cat <- readJPEG('cat.jpeg')
class(cat)
```

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



```
dim(cat)
```


```
## [1] 1365 2048 3
```

~~1365 2048 3~~

476 711 3

## 2. rgb 데이터 분할 및 주성분 분석

```
r <- cat[, , 1] # array에서 r에 해당하는 데이터
g <- cat[, , 2] # array에서 g에 해당하는 데이터
b <- cat[, , 3] # array에서 b에 해당하는 데이터
```

→ **표준화 안 한다는 옵션** 

```
cat.r.pca <- prcomp(r, center = F) # r 데이터 주성분분석
cat.g.pca <- prcomp(g, center = F) # g 데이터 주성분분석
cat.b.pca <- prcomp(b, center = F) # b 데이터 주성분분석
```

```
rgb.pca <- list(cat.r.pca, cat.g.pca, cat.b.pca) # 분석 결과 rgb로 합침
```



### 3. 차원축소하여 jpg로 저장

```
pc <- c(2,10,50,100,300) # 축소할 차원 수
for (i in pc) {
  pca.img <- sapply(rgb.pca, function(j) {
    compressed.img <- j$x[,1:i] %*% t(j$rotation[,1:i])
  }, simplify = 'array')
  writeJPEG(pca.img, paste('cat_pca_', i, '.jpeg', sep = ''))
}
```

→ **적용 행렬곱** → **rgb.PCA**  
 → **row의 값** → **eigenvector**  
 → **파일 이름 작성** → **행렬을** → **찍어쓰기 안 함**  
 → **적용을 다 한 뒤에 array로 만들어주는 의미**



↳ 고양이를 알아볼  
 정도라면 하고 싶다  
 ⇒ **30개**