PR4_201823869_조성우

조성우

2020 4 10

1.Factor

1.1. 명목형 자료 만들기

```
score <- factor(c(3,2,3,4,3,1,1,3,2,2,1,1,5),
                levels=c(1,2,3,4),
                labels=c("A","B","C","D"),
                ordered=T)
score
```

```
##  [1] C    B    C    D    C    A    A    C    B    B    A    A    <NA>
## Levels: A < B < C < D
```

1.2. 명목형 자료로 변환하기

```
# 문자를 벡터에 입력하였을때
fac_char <-c("포도","키위","메론","바나나","딸기")
attributes(fac_char)
```

```
## NULL
```

```
#문자벡터를 명목형 자료로 변환하였을때
fac_char = as.factor(fac_char)
attributes(fac_char)
```

```
## $levels
## [1] "딸기"   "메론"   "바나나" "키위"   "포도"
##
## $class
## [1] "factor"
```

1.3. 팩터형 자료 빈도 파악

```
table(score)
```

```
## score
## A B C D
## 4 3 4 1
```

```r
# 빈도가 3 이상인 데이터만 출력
tb <- table(score)
tb[tb>=3]
```

```
## score
## A B C
## 4 3 4
```

1.4. 서수형 자료와 명목형 자료의 차이

```r
score[1] > score[3] #(1)
```

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

```r
fac_char[1] > fac_char[2] #(2)
```

```
## Warning in Ops.factor(fac_char[1], fac_char[2]): '>' not meaningful for
factors
```

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

 # (1)은 서수형 자료형간의 비교인데 , 서수형 자료형이란 변수가 어떤 기준에 따라 순서판별이 가능한 자료형을 뜻하므로 두 자료간의 비교값이 T/F 의 논리형으로 출력되는것이고

 # (2)는 명목형 자료형간의 비교로, 명목형 자료형이란 크기나 순서가 의미가없고 이름만 의미를 부여할 수 있어 변수간 순서판별이 불가능한 자료형을 뜻하므로 비교값을 출력할 수 없다.

2. Matrix

2.1. matrix 생성

```r
mat <- matrix(1:8, nrow=2 , ncol=4, byrow=T)
dim(mat) ; length(mat) #dim 함수는 행, 열 순으로 차원을 출력
```

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

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

```r
matrix(1:8, nrow=2, ncol=4, byrow=F)
```

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

```r
matrix(1:8, nrow=2) #1~8의 수로 2 행 배열
```

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

```r
matrix(1:8,ncol=2) #1~8의 수로 2  열 배열
```

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

```r
matrix(1:8,ncol=4,byrow=T)
```

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

```r
matrix(1:9, nrow=3, ncol=3,
       dimnames = (list(c("r1", "r2", "r3"), c("c1","c2","c3"))))
```

```
##    c1 c2 c3
## r1  1  4  7
## r2  2  5  8
## r3  3  6  9
```

2.2. matrix 각 차원에 이름 부여

```r
mat
```

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

```r
rownames(mat) <- c("행 1","행 2")
colnames(mat) <- c("열 1","열 2","열 3","열 4")
mat
```

```
##      열 1 열 2 열 3 열 4
## 행 1    1    2    3    4
## 행 2    5    6    7    8
```

2.3. rbind()와 cbind()를 이용한 매트릭스 생성

```r
x <- 2:4 ;x
```

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

```r
y <- 9:11 ;y
```

```
## [1]  9 10 11
```

```r
cbind(x,y)
```

```
##      x  y
## [1,] 2  9
## [2,] 3 10
## [3,] 4 11
```

```
rbind(x,y)
```

```
##   [,1] [,2] [,3]
## x    2    3    4
## y    9   10   11
```

2.4. rbind()와cbind()를 사용한 데이터 추가

```
mat
```

```
##      열 1 열 2 열 3 열 4
## 행 1    1    2    3    4
## 행 2    5    6    7    8
```

```
cbind(mat, 10:11)
```

```
##      열 1 열 2 열 3 열 4
## 행 1    1    2    3    4 10
## 행 2    5    6    7    8 11
```

```
rbind(mat,20:23)
```

```
##      열 1 열 2 열 3 열 4
## 행 1    1    2    3    4
## 행 2    5    6    7    8
##        20   21   22   23
```

2.5. matrix 데이터 접근과 변환

```
x <- 1:3 ;x
```

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

```
y <- 10:12 ; y
```

```
## [1] 10 11 12
```

```
mat <-cbind(x,y) ; mat
```

```
##      x  y
## [1,] 1 10
## [2,] 2 11
## [3,] 3 12
```

```
mat[1,1] <- 100 ; mat
```

```
##        x  y
## [1,] 100 10
## [2,]   2 11
## [3,]   3 12
```

```
mat[2,] <- mat[2,] / 4 ; mat
```

```
##          x     y
## [1,] 100.0 10.00
## [2,]   0.5  2.75
## [3,]   3.0 12.00
```

```
mat[,2] <- mat[,2] - mat[,1]*3 ; mat
```

```
##          x       y
## [1,] 100.0 -290.00
## [2,]   0.5    1.25
## [3,]   3.0    3.00
```

3.List

3.1. 여러 벡터를 이용해 리스트 만들기

```
str_vec <- c("korea","USA", "Japan") #문자열 벡터
num_vec <- c(100,200,300,400,500) #숫자 벡터
mat <- matrix(2:9,2,4)
list_tot <- list(str_vec,num_vec,mat)
print(list_tot)
```

```
## [[1]]
## [1] "korea" "USA"   "Japan"
##
## [[2]]
## [1] 100 200 300 400 500
##
## [[3]]
##      [,1] [,2] [,3] [,4]
## [1,]    2    4    6    8
## [2,]    3    5    7    9
```

```
names(list_tot) <- c('str_vec','num_vec','mat');list_tot
```

```
## $str_vec
## [1] "korea" "USA"   "Japan"
##
## $num_vec
## [1] 100 200 300 400 500
##
## $mat
```

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

3.2. list 함수 내에서 성분의 이름 지정하여 리스트 만들기

```
list_tot2 = list(seq = seq(1,10,2),
                 str = c("토끼","사자","코끼리","양"),
                 plus = rep(c('고구마','감자','옥수수'),2))
print(list_tot2)
```

```
## $seq
## [1] 1 3 5 7 9
##
## $str
## [1] "토끼"   "사자"   "코끼리" "양"
##
## $plus
## [1] "고구마" "감자"   "옥수수" "고구마" "감자"   "옥수수"
```

3.3. 데이터의 속성을 확인하는 다양한 함수

```
class(list_tot)
```

```
## [1] "list"
```

```
length(list_tot)
```

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

```
attributes(list_tot)
```

```
## $names
## [1] "str_vec" "num_vec" "mat"
```

```
str(list_tot)
```

```
## List of 3
##  $ str_vec: chr [1:3] "korea" "USA" "Japan"
##  $ num_vec: num [1:5] 100 200 300 400 500
##  $ mat    : int [1:2, 1:4] 2 3 4 5 6 7 8 9
```

3.4. 리스트의 성분에 접근하기

```
# [ ] 연산자 또는 $ 연산자를 활용해 추출


list_tot2[1] #첫번째 성분
```

```
## $seq
## [1] 1 3 5 7 9
```

```
list_tot2[3]
```

```
## $plus
## [1] "고구마" "감자"   "옥수수" "고구마" "감자"   "옥수수"
```

```
list_tot2[1:2]
```

```
## $seq
## [1] 1 3 5 7 9
##
## $str
## [1] "토끼"   "사자"   "코끼리" "양"
```

```
list_tot2$seq #'seq'라는 성분
```

```
## [1] 1 3 5 7 9
```

```
list_tot2$str
```

```
## [1] "토끼"   "사자"   "코끼리" "양"
```

3.5. 리스트의 성분인에 있는 원소에 접근하기

```
#[[]]연산자 또는 $ 연산자와 []로 추출
list_tot[[3]][1] #3번째 성분의 첫번째 원소
```

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

```
list_tot2$seq[3] #seq 성분의 세번째 원소
```

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

```
list_tot2$str[1:2]
```

```
## [1] "토끼" "사자"
```

3.6 리스트의 성분이나 원소 조작하기

```
#성분이나 원소 삭제 또는 추가하기


list_tot[1] <- NULL #첫번째 성분 삭제
str(list_tot)
```

```
## List of 2
##  $ num_vec: num [1:5] 100 200 300 400 500
##  $ mat    : int [1:2, 1:4] 2 3 4 5 6 7 8 9
```

```
list_tot2$str[1] <- "고양이" #str 성분 첫번째 원소 덮어쓰기
str(list_tot2)
```

```
## List of 3
##  $ seq : num [1:5] 1 3 5 7 9
##  $ str : chr [1:4] "고양이" "사자" "코끼리" "양"
##  $ plus: chr [1:6] "고구마" "감자" "옥수수" "고구마" ...
```

```
list_tot$NEW <- 2:5 #새로운 성분 추가
str(list_tot)
```

```
## List of 3
##  $ num_vec: num [1:5] 100 200 300 400 500
##  $ mat    : int [1:2, 1:4] 2 3 4 5 6 7 8 9
##  $ NEW    : int [1:4] 2 3 4 5
```

3.7. 리스트의 성분에 하위 리스트 추가하여 계층적으로 리스트 만들기

```
list_tot$hierarchy[[1]] <- list_tot2 #리스트의 hierarchy 라는 성분에 list_tot2 를 할당
str(list_tot)
```

```
## List of 4
##  $ num_vec  : num [1:5] 100 200 300 400 500
##  $ mat      : int [1:2, 1:4] 2 3 4 5 6 7 8 9
##  $ NEW      : int [1:4] 2 3 4 5
##  $ hierarchy:List of 1
##   ..$ :List of 3
##   .. ..$ seq : num [1:5] 1 3 5 7 9
##   .. ..$ str : chr [1:4] "고양이" "사자" "코끼리" "양"
##   .. ..$ plus: chr [1:6] "고구마" "감자" "옥수수" "고구마" ...
```

4. Array

4.1. Array 생성하기

```
# array 함수로 array 생성하기
arr <- array(1:18, dim=c(3,3,2),
             dimnames=list(c("KOR","CHI","JAP"),
                           c("GDP.R","USD.R","Cuur.Acc"),
                           c("2011Y","2012Y")))
arr
```

```
## , , 2011Y
##
##     GDP.R USD.R Cuur.Acc
## KOR     1     4        7
## CHI     2     5        8
## JAP     3     6        9
```

```
##
## , , 2012Y
##
##     GDP.R USD.R Cuur.Acc
## KOR    10    13       16
## CHI    11    14       17
## JAP    12    15       18
```

```
# 벡터 생성후 차원을 배열하여 array로 변환하기
arr1 <- 1:18
dim(arr1) <- c(3,3,2)
dimnames(arr1) <-list(c("KOR","CHI","JAP"),
                      c("GDP.R","USD.R","Cuur.Acc"),
                      c("2011Y","2012Y"))
arr1
```

```
## , , 2011Y
##
##     GDP.R USD.R Cuur.Acc
## KOR     1     4        7
## CHI     2     5        8
## JAP     3     6        9
##
## , , 2012Y
##
##     GDP.R USD.R Cuur.Acc
## KOR    10    13       16
## CHI    11    14       17
## JAP    12    15       18
```

```
arr1 == arr #앞에서 만든 배열과 같은지 비교교
```

```
## , , 2011Y
##
##     GDP.R USD.R Cuur.Acc
## KOR  TRUE  TRUE     TRUE
## CHI  TRUE  TRUE     TRUE
## JAP  TRUE  TRUE     TRUE
##
## , , 2012Y
##
##     GDP.R USD.R Cuur.Acc
## KOR  TRUE  TRUE     TRUE
## CHI  TRUE  TRUE     TRUE
## JAP  TRUE  TRUE     TRUE
```

4.2. Array 조작 방법

4.2.1 [,,] 인덱싱으로 각 원소에 접근하기

```
arr
```

```
## , , 2011Y
##
##     GDP.R USD.R Cuur.Acc
## KOR     1     4        7
## CHI     2     5        8
## JAP     3     6        9
##
## , , 2012Y
##
##     GDP.R USD.R Cuur.Acc
## KOR    10    13       16
## CHI    11    14       17
## JAP    12    15       18
```

```
arr[1,,]
```

```
##          2011Y 2012Y
## GDP.R        1    10
## USD.R        4    13
## Cuur.Acc     7    16
```

```
arr[,-2,] #3 개국의 GDP.R 와 Cuur.Acc
```

```
## , , 2011Y
##
##     GDP.R Cuur.Acc
## KOR     1        7
## CHI     2        8
## JAP     3        9
##
## , , 2012Y
##
##     GDP.R Cuur.Acc
## KOR    10       16
## CHI    11       17
## JAP    12       18
```

```
arr[,,2]#3 개국의 2012 년 자료
```

```
##     GDP.R USD.R Cuur.Acc
## KOR    10    13       16
## CHI    11    14       17
## JAP    12    15       18
```

```
arr[,,'2012Y']#이름으로 추출(3개국의 2012년 자료)

##     GDP.R USD.R Cuur.Acc
## KOR    10    13       16
## CHI    11    14       17
## JAP    12    15       18

arr[c(T,T,F),,2]#한국, 중국의 2012년 자료 -  일본 제외

##     GDP.R USD.R Cuur.Acc
## KOR    10    13       16
## CHI    11    14       17

arr[-2,,2] #한국, 일본의 2012년 자료 -  중국 제외

##     GDP.R USD.R Cuur.Acc
## KOR    10    13       16
## JAP    12    15       18
```

4.2.2. 배열원소의 추출 및 수정

```
arr.tmp <- arr
arr.tmp

## , , 2011Y
##
##     GDP.R USD.R Cuur.Acc
## KOR     1     4        7
## CHI     2     5        8
## JAP     3     6        9
##
## , , 2012Y
##
##     GDP.R USD.R Cuur.Acc
## KOR    10    13       16
## CHI    11    14       17
## JAP    12    15       18

arr.tmp[,,1] <-c(5,6,4)
arr.tmp

## , , 2011Y
##
##     GDP.R USD.R Cuur.Acc
## KOR     5     5        5
## CHI     6     6        6
## JAP     4     4        4
##
## , , 2012Y
##
##     GDP.R USD.R Cuur.Acc
```

```
## KOR    10    13       16
## CHI    11    14       17
## JAP    12    15       18

arr.tmp[,1,2] <-NA
arr.tmp

## , , 2011Y
##
##     GDP.R USD.R Cuur.Acc
## KOR     5     5        5
## CHI     6     6        6
## JAP     4     4        4
##
## , , 2012Y
##
##     GDP.R USD.R Cuur.Acc
## KOR    NA    13       16
## CHI    NA    14       17
## JAP    NA    15       18

arr.tmp[is.na(arr.tmp)] <- c(8,5,2)
arr.tmp

## , , 2011Y
##
##     GDP.R USD.R Cuur.Acc
## KOR     5     5        5
## CHI     6     6        6
## JAP     4     4        4
##
## , , 2012Y
##
##     GDP.R USD.R Cuur.Acc
## KOR     8    13       16
## CHI     5    14       17
## JAP     2    15       18
```

5. 기타

5.1 NA값 다루기

```
# is.na 함수와 complete.cases  하무를 사용해 결측값 파악하기

x <- c(1,2,NA,4,NA,5)
is.na(x) #NA 값 여부에 대한 논리 판단 결과

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

```r
bad <- is.na(x) # 벡터 x의 na값 여부에 대한 논리판단결과를 bad에 할당
y <- x[!bad] # bad에 할당된 논리판단결과를 역으로 바꾸고 T에 해당하는 항과 같은 위치의 x의 항을 y에 할당( 결과적으로 na가 배제됨)
mean(y) #그 값의 평균산출
```

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

```r
x <- c(1,2,NA,4,NA,5)
good <- complete.cases(x)
x[good]
```

```
## [1] 1 2 4 5
```

#PR4 연습문제###### 다음은2015~2020년 대한민국 경제활동인구에 관한 통계 중 실업률에 관한 내용이다. ###### 남성실업률: 3.8 3.4 3.2 3.1 3.8 3.2 3.2 2.9 3,9 3,6 3.4 3.3 4.0 3.8 3.6 3.2 4.3 3.8 3.7 3.2 4.2 4.0 3.7 3.4 ###### 여성실업률: 3.7 3.0 2.7 2.5 3.3 2.9 2.6 2.6 4.0 3.7 3.1 3.1 4.2 3.8 3.2 3.0 4.2 3.6 3.3 3.2 4.4 3.7 3.2 2.9 ###### 실업률: (남성실업률+여성실업률)/2

문제1.

위를 참고하여 아래와 같은list를 만들어 보세요.

list의 이름은unemploy_rate_list로 지정해 주세요.

남/여성 전체 실업률을 구해서 추가해보세요.

```r
country = "korea"
start = 2015
end = 2020
type = "quarterly"
M_UR = c(3.8, 3.4, 3.2, 3.1, 3.8, 3.2, 3.2, 2.9, 3.9, 3.6, 3.4, 3.3, 4.0,
3.8, 3.6, 3.2, 4.3, 3.8, 3.7, 3.2, 4.2, 4.0, 3.7, 3.4) #남성실업률
F_UR = c(3.7, 3.0, 2.7, 2.5, 3.3, 2.9, 2.6, 2.6, 4.0, 3.7, 3.1, 3.1, 4.2,
3.8, 3.2, 3.0, 4.2, 3.6, 3.3, 3.2, 4.4, 3.7, 3.2, 2.9) #여성실업률


unemploy_rate_list <- list(country ,start ,end ,type ,M_UR ,F_UR)
names(unemploy_rate_list) <- c("country","start","end","type","M_UR","F_UR")


total_UR  <- numeric(24) #총 실업률 변수선언 및 총실업률 계산
for (i in 1:24){
 total_UR[i]  <- ( M_UR[i] + F_UR[i]) / 2
```

```
}
total_UR
```

```
##  [1] 3.75 3.20 2.95 2.80 3.55 3.05 2.90 2.75 3.95 3.65 3.25 3.20 4.10 3.80
3.40
## [16] 3.10 4.25 3.70 3.50 3.20 4.30 3.85 3.45 3.15
```

```
unemploy_rate_list$total_UR <- total_UR;unemploy_rate_list
#unemploy_rate_list에 전체실업률 성분 추가 후 출력
```

```
## $country
## [1] "korea"
##
## $start
## [1] 2015
##
## $end
## [1] 2020
##
## $type
## [1] "quarterly"
##
## $M_UR
##  [1] 3.8 3.4 3.2 3.1 3.8 3.2 3.2 2.9 3.9 3.6 3.4 3.3 4.0 3.8 3.6 3.2 4.3
3.8 3.7
## [20] 3.2 4.2 4.0 3.7 3.4
##
## $F_UR
##  [1] 3.7 3.0 2.7 2.5 3.3 2.9 2.6 2.6 4.0 3.7 3.1 3.1 4.2 3.8 3.2 3.0 4.2
3.6 3.3
## [20] 3.2 4.4 3.7 3.2 2.9
##
## $total_UR
##  [1] 3.75 3.20 2.95 2.80 3.55 3.05 2.90 2.75 3.95 3.65 3.25 3.20 4.10 3.80
3.40
## [16] 3.10 4.25 3.70 3.50 3.20 4.30 3.85 3.45 3.15
```

문제2.

문제1에서 만든 리스트를 다음과 같이 자유롭게 조작해보세요. (성분삭제하기, 원소덮어쓰기, 성분추가하기)

각 조작한 내용은 조작후 결과를 모두 출력해주세요.

```
unemploy_rate_list[c(1,3)] <- NULL;unemploy_rate_list #country와 end 성분 삭제
```

```
## $start
## [1] 2015
##
## $type
```

```
## [1] "quarterly"
##
## $M_UR
##  [1] 3.8 3.4 3.2 3.1 3.8 3.2 3.2 2.9 3.9 3.6 3.4 3.3 4.0 3.8 3.6 3.2 4.3
3.8 3.7
## [20] 3.2 4.2 4.0 3.7 3.4
##
## $F_UR
##  [1] 3.7 3.0 2.7 2.5 3.3 2.9 2.6 2.6 4.0 3.7 3.1 3.1 4.2 3.8 3.2 3.0 4.2
3.6 3.3
## [20] 3.2 4.4 3.7 3.2 2.9
##
## $total_UR
##  [1] 3.75 3.20 2.95 2.80 3.55 3.05 2.90 2.75 3.95 3.65 3.25 3.20 4.10 3.80
3.40
## [16] 3.10 4.25 3.70 3.50 3.20 4.30 3.85 3.45 3.15
```

```
unemploy_rate_list$start[1] <- 2020;unemploy_rate_list  #원소 덮어쓰기 2015 ->
2020
```

```
## $start
## [1] 2020
##
## $type
## [1] "quarterly"
##
## $M_UR
##  [1] 3.8 3.4 3.2 3.1 3.8 3.2 3.2 2.9 3.9 3.6 3.4 3.3 4.0 3.8 3.6 3.2 4.3
3.8 3.7
## [20] 3.2 4.2 4.0 3.7 3.4
##
## $F_UR
##  [1] 3.7 3.0 2.7 2.5 3.3 2.9 2.6 2.6 4.0 3.7 3.1 3.1 4.2 3.8 3.2 3.0 4.2
3.6 3.3
## [20] 3.2 4.4 3.7 3.2 2.9
##
## $total_UR
##  [1] 3.75 3.20 2.95 2.80 3.55 3.05 2.90 2.75 3.95 3.65 3.25 3.20 4.10 3.80
3.40
## [16] 3.10 4.25 3.70 3.50 3.20 4.30 3.85 3.45 3.15
```

```
unemploy_rate_list$seq <- seq(50,100,2);unemploy_rate_list #성분 추가하기
```

```
## $start
## [1] 2020
##
## $type
## [1] "quarterly"
##
## $M_UR
```

```
##  [1] 3.8 3.4 3.2 3.1 3.8 3.2 3.2 2.9 3.9 3.6 3.4 3.3 4.0 3.8 3.6 3.2 4.3
3.8 3.7
## [20] 3.2 4.2 4.0 3.7 3.4
##
## $F_UR
##  [1] 3.7 3.0 2.7 2.5 3.3 2.9 2.6 2.6 4.0 3.7 3.1 3.1 4.2 3.8 3.2 3.0 4.2
3.6 3.3
## [20] 3.2 4.4 3.7 3.2 2.9
##
## $total_UR
##  [1] 3.75 3.20 2.95 2.80 3.55 3.05 2.90 2.75 3.95 3.65 3.25 3.20 4.10 3.80
3.40
## [16] 3.10 4.25 3.70 3.50 3.20 4.30 3.85 3.45 3.15
##
## $seq
##  [1]  50  52  54  56  58  60  62  64  66  68  70  72  74  76  78  80  82
84  86
## [20]  88  90  92  94  96  98 100
```

문제3.

위를 참고하여 아래와 같은array를 만들어 보세요.

array 이름은unemploy_rate_arr 로 지정해주세요.

Hint: 1.남성2.여성3.전체실업률

```
country = "korea"
start = 2015
end = 2020
type = "quarterly"
M_UR = c(3.8, 3.4, 3.2, 3.1, 3.8, 3.2, 3.2, 2.9, 3.9, 3.6, 3.4, 3.3, 4.0,
3.8, 3.6, 3.2, 4.3, 3.8, 3.7, 3.2, 4.2, 4.0, 3.7, 3.4) #남성실업률
F_UR = c(3.7, 3.0, 2.7, 2.5, 3.3, 2.9, 2.6, 2.6, 4.0, 3.7, 3.1, 3.1, 4.2,
3.8, 3.2, 3.0, 4.2, 3.6, 3.3, 3.2, 4.4, 3.7, 3.2, 2.9) #여성실업률


unemploy_rate_list <- list(country ,start ,end ,type ,M_UR ,F_UR)
names(unemploy_rate_list) <- c("country","start","end","type","M_UR","F_UR")



total_UR  <- numeric(24) #총 실업률 변수선언 및 총실업률 계산
for (i in 1:24){
 total_UR[i]  <- ( M_UR[i] + F_UR[i]) / 2
}
total_UR
```

```
##  [1] 3.75 3.20 2.95 2.80 3.55 3.05 2.90 2.75 3.95 3.65 3.25 3.20 4.10 3.80
3.40
## [16] 3.10 4.25 3.70 3.50 3.20 4.30 3.85 3.45 3.15
```

```
unemploy_rate_list$total_UR <- total_UR;unemploy_rate_list
#unemploy_rate_list에 전체실업률 성분 추가 후 출력
```

```
## $country
## [1] "korea"
##
## $start
## [1] 2015
##
## $end
## [1] 2020
##
## $type
## [1] "quarterly"
##
## $M_UR
##  [1] 3.8 3.4 3.2 3.1 3.8 3.2 3.2 2.9 3.9 3.6 3.4 3.3 4.0 3.8 3.6 3.2 4.3
3.8 3.7
## [20] 3.2 4.2 4.0 3.7 3.4
##
## $F_UR
##  [1] 3.7 3.0 2.7 2.5 3.3 2.9 2.6 2.6 4.0 3.7 3.1 3.1 4.2 3.8 3.2 3.0 4.2
3.6 3.3
## [20] 3.2 4.4 3.7 3.2 2.9
##
## $total_UR
##  [1] 3.75 3.20 2.95 2.80 3.55 3.05 2.90 2.75 3.95 3.65 3.25 3.20 4.10 3.80
3.40
## [16] 3.10 4.25 3.70 3.50 3.20 4.30 3.85 3.45 3.15
```

```
#-------------------------------------------------------------------------
----------------
```

```
unemploy_rate_arr <- c(M_UR,F_UR,total_UR);unemploy_rate_arr
```

```
##  [1] 3.80 3.40 3.20 3.10 3.80 3.20 3.20 2.90 3.90 3.60 3.40 3.30 4.00 3.80
3.60
## [16] 3.20 4.30 3.80 3.70 3.20 4.20 4.00 3.70 3.40 3.70 3.00 2.70 2.50 3.30
2.90
## [31] 2.60 2.60 4.00 3.70 3.10 3.10 4.20 3.80 3.20 3.00 4.20 3.60 3.30 3.20
4.40
## [46] 3.70 3.20 2.90 3.75 3.20 2.95 2.80 3.55 3.05 2.90 2.75 3.95 3.65 3.25
3.20
## [61] 4.10 3.80 3.40 3.10 4.25 3.70 3.50 3.20 4.30 3.85 3.45 3.15
```

```
dim(unemploy_rate_arr) <-c(4,6,3)
dimnames(unemploy_rate_arr) =
list(c("1/4","2/4","3/4","4/4"),c('2015Y','2016Y','2017Y','2018Y','2019Y','20
20Y'),c('man_unemp','women_unemp','total_unemp'))
unemploy_rate_arr
```

```
## , , man_unemp
##
##     2015Y 2016Y 2017Y 2018Y 2019Y 2020Y
## 1/4   3.8   3.8   3.9   4.0   4.3   4.2
## 2/4   3.4   3.2   3.6   3.8   3.8   4.0
## 3/4   3.2   3.2   3.4   3.6   3.7   3.7
## 4/4   3.1   2.9   3.3   3.2   3.2   3.4
##
## , , women_unemp
##
##     2015Y 2016Y 2017Y 2018Y 2019Y 2020Y
## 1/4   3.7   3.3   4.0   4.2   4.2   4.4
## 2/4   3.0   2.9   3.7   3.8   3.6   3.7
## 3/4   2.7   2.6   3.1   3.2   3.3   3.2
## 4/4   2.5   2.6   3.1   3.0   3.2   2.9
##
## , , total_unemp
##
##     2015Y 2016Y 2017Y 2018Y 2019Y 2020Y
## 1/4  3.75  3.55  3.95   4.1  4.25  4.30
## 2/4  3.20  3.05  3.65   3.8  3.70  3.85
## 3/4  2.95  2.90  3.25   3.4  3.50  3.45
## 4/4  2.80  2.75  3.20   3.1  3.20  3.15
```

문제4.

문제3에서 만든array에서2017Y의 자료를 제외한 남성, 여성, 총 실업률을 출력해보세요

문제3에서 만든array에서 총 실업률만 출력해보세요

```
print("2017년도 제외 남성, 여성,  총 실업률 출력")
```

```
## [1] "2017년도 제외 남성, 여성,  총 실업률 출력"
```

```
a <- unemploy_rate_arr[,-3,];a#2017년도 제외 남성, 여성,  총 실업률 출력
```

```
## , , man_unemp
##
##     2015Y 2016Y 2018Y 2019Y 2020Y
## 1/4   3.8   3.8   4.0   4.3   4.2
## 2/4   3.4   3.2   3.8   3.8   4.0
## 3/4   3.2   3.2   3.6   3.7   3.7
## 4/4   3.1   2.9   3.2   3.2   3.4
```

```
##
## , , women_unemp
##
##     2015Y 2016Y 2018Y 2019Y 2020Y
## 1/4   3.7   3.3   4.2   4.2   4.4
## 2/4   3.0   2.9   3.8   3.6   3.7
## 3/4   2.7   2.6   3.2   3.3   3.2
## 4/4   2.5   2.6   3.0   3.2   2.9
##
## , , total_unemp
##
##     2015Y 2016Y 2018Y 2019Y 2020Y
## 1/4  3.75  3.55   4.1  4.25  4.30
## 2/4  3.20  3.05   3.8  3.70  3.85
## 3/4  2.95  2.90   3.4  3.50  3.45
## 4/4  2.80  2.75   3.1  3.20  3.15
```

```
print("총실업률만 출력")
```

```
## [1] "총실업률만 출력"
```

```
b <- unemploy_rate_arr[,,3];b #총실업률만 출력
```

```
##     2015Y 2016Y 2017Y 2018Y 2019Y 2020Y
## 1/4  3.75  3.55  3.95   4.1  4.25  4.30
## 2/4  3.20  3.05  3.65   3.8  3.70  3.85
## 3/4  2.95  2.90  3.25   3.4  3.50  3.45
## 4/4  2.80  2.75  3.20   3.1  3.20  3.15
```

PR 4 도전문제

######다음은 가우스 조던법을 이용하여 연립방정식의 해를 구하는 방법을 서술한 것입니다. 문제에 대한 풀이를 작성할 시에 과정을 통해 도출되는 matrix의 결과값 외의 설명은 출력될 필요는 없습니다.

문제1.

```
#과정1
print("과정1")
```

```
## [1] "과정1"
```

```
mat0<-matrix(c(2,3,1,4,4,1,-3,-2,-1,2,2,2),nrow=3,ncol=4,byrow=T);mat0
```

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

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

```
#과정2
print("과정2")
```

```
## [1] "과정2"
```

```
mat0[1,]<-mat0[1,]/2;mat0
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1  1.5  0.5    2
## [2,]    4  1.0 -3.0   -2
## [3,]   -1  2.0  2.0    2
```

```
#과정3
print("과정3")
```

```
## [1] "과정3"
```

```
mat1 = mat0
mat1[2,] <- mat0[2,] - mat0[1,]*4
mat1[3,] <- mat0[3,] - mat0[1,]*(-1) ;mat1
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1  1.5  0.5    2
## [2,]    0 -5.0 -5.0  -10
## [3,]    0  3.5  2.5    4
```

```
#과정4
print("과정4")
```

```
## [1] "과정4"
```

```
mat1[2,] <- mat1[2,]/-5 ;mat1
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1  1.5  0.5    2
## [2,]    0  1.0  1.0    2
## [3,]    0  3.5  2.5    4
```

```
#과정5
print("과정5")
```

```
## [1] "과정5"
```

```
mat2 = mat1
mat2[1,] <- mat1[1,] - mat1[2,] * 1.5
mat2[3,] <- mat1[3,] - mat1[2,] * 3.5;mat2
```

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

```
#과정6
print("과정6")
```

```
## [1] "과정6"
```

```
mat2[3,] <- mat2[3,]/-1;mat2
```

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

```
#과정7
print("과정7")
```

```
## [1] "과정7"
```

```
mat3 = mat2
mat3[1,] <- mat2[1,] - mat2[3,] * (-1)
mat3[2,] <- mat2[2,] - mat2[3,] * 1;mat3
```

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

```
cat("해 x: ",mat3[1,4],",y: ",mat3[2,4],",z: ",mat3[3,4])
```

```
## 해 x:  2 ,y:  -1 ,z:  3
```

문제2.

```
# 짝수번호의 과정은 그대로하되, 홀수번호(3~부터) 의 과정에 제시된 %*% 의 행렬곱셈과 항등행렬 변형의
활용을 적용시켰다.

matz <- matrix(c(1,0,0,0,1,0,0,0,1),3,3);matz #Hint 에서 제시한 항등행렬 matz 생성
```

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

```
matx = matz  #홀수번호(3~)의 과정마다 항등행렬이 새로 사용될것을 대비하여 편의를 위해 matx에 그대로 입력
```

```
#과정1
print("과정1")
```

```
## [1] "과정1"
```

```
mat0<-matrix(c(2,3,1,4,4,1,-3,-2,-1,2,2,2),nrow=3,ncol=4,byrow=T);mat0
```

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

```
#과정2
print("과정2")
```

```
## [1] "과정2"
```

```
mat0[1,]<-mat0[1,]/2;mat0
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1  1.5  0.5    2
## [2,]    4  1.0 -3.0   -2
## [3,]   -1  2.0  2.0    2
```

```
#과정3
print("과정3")
```

```
## [1] "과정3"
```

```
matx[2,1] <- matx[1,1]*-4
matx[3,1] <- matx[1,1]*-(-1)
```

```
mat0 <- matx %*% mat0;mat0
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1  1.5  0.5    2
## [2,]    0 -5.0 -5.0  -10
## [3,]    0  3.5  2.5    4
```

```r
#과정4
print("과정4")

## [1] "과정4"

mat0[2,] <- mat0[2,] / -5;mat0

##      [,1] [,2] [,3] [,4]
## [1,]    1  1.5  0.5    2
## [2,]    0  1.0  1.0    2
## [3,]    0  3.5  2.5    4

#과정5
print("과정5")

## [1] "과정5"

matx = matz
matx[1,2] <- matx[2,2]*-1.5
matx[3,2] <- matx[2,2]*-3.5

mat0 <- matx %*% mat0;mat0

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

#과정6
print("과정6")

## [1] "과정6"

mat0[3,] <- mat0[3,]/-1;mat0

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

#과정7
print("과정7")

## [1] "과정7"

matx = matz
matx[1,3] <- matx[3,3]*-(-1)
```

```r
matx[2,3] <- matx[3,3]*-1

mat0 <- matx %*% mat0;mat0

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