apply 함수

- 개요
- 계산 보기
- 참고

개요

적용

행렬이나 데이터 프레임 및 배열 등에서 행별/열별 및 특정한 인덱스별로 함수를 적용하고자 할 때 for 루프 등을 사용한 번거로운 계산을 대체할 수 있는 함수

사용 함수

```
apply(X, MARGIN, FUN, ...)
```

매개변수

- X : 행렬 또는 배열로 FUN에 설정한 함수를 적용할 자료
- MARGIN : 몇 번째 배열인지 설정. 행렬이나 데이터 프레임의 경우 1이면 행, 2는 열
- FUN : 적용할 함수를 설정하며 사용자가 만든 함수 또는 R의 내장함수
- ... : FUN에 설정된 함수가 매개변수를 필요로 할 때 그 매개변수를 설정

사용할 자료

```
> score <- read.table(file="c:/hwp/STI/score.txt", header = TRUE,
          fileEncoding="UTF-8",  stringsAsFactors=FALSE)
```

50명 학생의 국어, 영어, 수학, 과학 및 사회5개 과목별 점수를 저장한 텍스트 파일을 읽어옴
※ 다음의 링크를 넣으면 볼 수 있음(c:/hwp/STI/score.txt)

- 첫 6줄의 자료

```
> head(score)
      이름 국어 영어 수학 과학 사회
1 한대성   67   87   60   85   80
2 강준호   63   73   82   73   85
3 김종욱   74   53   76   72   58
4 박상호   55   65   49   84   96
5 김소현   76   69   69   92   70
      ⋮
```

계산 보기

apply 함수로 계산하기

자료

앞의 데이터 프레임 score에 저장된 5개 주요과목의 점수에서 50명의 개별 학생별 점수의 합을 계산해보자.

```
> apply(score[,-1], 1, sum)
[1] 379 376 333 349 376 384 371 333 329 350 337 368 364 366 352 345 374 350
[19] 337 354 361 386 351 384 321 371 388 405 349 372 346 312 409 378 359 365
[37] 369 367 392 348 409 377 364 408 358 364 357 391 357 353
```

각 과목별 평균점수의 계산

```
> apply(score[,-1], 2, mean)
국어    영어    수학    과학    사회
67.52  75.36  65.74  79.86  75.48
```

계산 보기

apply 함수로 계산하기

추후에 사용하기 위해 앞에서 계산한 학생별 점수를 원래의 데이터 프레임에 붙이기

```
total <- apply(score[,-1], 1, sum)
score <- cbind(score, total)
head(score)
```

	이름	국어	영어	수학	과학	사회	total
1	한대성	67	87	60	85	80	379
2	강준호	63	73	82	73	85	376
3	김종욱	74	53	76	72	58	333
4	박상호	55	65	49	84	96	349
5	김소현	76	69	69	92	70	376
6	김태성	59	98	70	77	80	384

계산 보기

apply 함수로 계산하기

과목별 평균 및 전체 평균을 score의 마지막 줄에 포함하기

```
> average <- apply(score[,-1], 2, mean)
> score <- rbind(score, c("평균", average))
> tail(score)
```

	이름	국어	영어	수학	과학	사회	total
46	성홍주	50	77	74	83	80	364
47	정선경	84	82	60	61	70	357
48	최수진	76	86	79	73	77	391
49	최지영	71	70	40	91	85	357
50	김민호	56	86	58	70	83	353
51	평균	67.52	75.36	65.74	79.86	75.48	363.96

참고

rbind 함수 또는 rbind.data.frame 함수를 사용하는 경우

데이터 프레임 내의 모든 값이 문자열로 바뀜

- 추후 이 score를 사용하여 연산을 할 때 문제가 생길 수 있으며 이를 위해 아래와 같은 명령으로 숫자로 변경하는 것이 좋음

```
> for (j in 2:dim(score)[2]) score[,j] <- as.numeric(score[,j])
```

lapply 함수

- 개요
- 계산 보기
- 참고

개요

적용

apply 함수가 데이터 프레임이나 배열 등에서 적용되는데 반해, lapply 함수는 자료가 list인 경우 list의 각 항목에 특정한 함수를 적용

사용 함수

```
lapply(X, FUN, ...)
```

매개변수

- X : 각각의 원소에 대해서 FUN에 설정한 함수를 적용함
  대개 X는 list를 원소로 갖는 경우에 사용함(list + apply = lapply)
- ... : FUN에 설정된 함수가 매개변수를 필요로 할 때 그 매개변수를 설정함

계산 보기

lapply 함수로 계산하기

자료

R 개체 x가 list일 때 lapply 함수를 적용해보자.

자료 x

```
> x <- list(a = 1:10, beta = exp(-3:3), logic = c(TRUE,FALSE,FALSE,TRUE))
```

- a : 1부터 10 사이의 자연수
- beta : $e^{-3}, e^{-2}, \ldots, e^{3}$
- logic : T, F, F, T의 논리 값

이 세 벡터를 포함한 list임

계산 보기

lapply 함수로 계산하기

x의 값 확인하기

```
> x
$a
[1] 1 2 3 4 5 6 7 8 9 10

$beta
[1] 0.04978707 0.13533528 0.36787944 1.00000000 2.71828183 7.38905610
[7] 20.08553692

$logic
[1] TRUE FALSE FALSE TRUE
```

계산 보기

lapply 함수로 계산하기

각 개체의 평균 계산하기

```
> lapply(x, mean)
$a
[1] 5.5

$beta
[1] 4.535125

$logic
[1] 0.5
```

참고

논리형값의 산술연산 : TRUE는 1, FALSE는 0으로 환산하여 계산함

함수에 매개변수가 필요한 경우 추가로 설정함

```
> lapply(x, quantile, probs = (1:3)/4)
```

- x의 각 개체에 대해서 quantile 함수 적용, probs는 quantile 함수의 매개변수

결과

```
$a
25%  50%  75%
3.25  5.50  7.75

$beta
       25%         50%         75%
0.2516074  1.0000000  5.0536690

$logic
25% 50% 75%
0.0  0.5  1.0
```

sapply 함수

- 개요
- 계산 보기

개요

적용

- 함수의 호출결과가 list 개체로 반환됨
- 함수의 호출결과로 추가 연산을 하거나 출력을 좀 더 가시적으로 하기 위해서는 출력이 list인 경우보다 행렬이나 벡터로 만들어지는 것이 나을 수 있는데 이를 위해 R의 내장함수에 sapply 함수가 제공됨

사용 함수

```
sapply(X, FUN, ...)
```

- 사용법은 lapply 함수와 같으나 lapply 함수의 결과를 벡터 또는 행렬로 만들어 주는 summary 기능을 제공하는 점만 다름

계산 보기

sapply 함수로 계산하기

자료

x에 대해서 lapply 대신 sapply를 호출해보자.

```
> sapply(x, mean)
       a         beta        logic
5.500000    4.535125    0.500000
```

계산 보기

sapply 함수로 계산하기

list의 각 항목 a, beta 및 logic에 대한 출력값이 한 개일 경우 벡터로 출력을 만들어주기

```
> sapply(x, quantile)
         a        beta   logic
0%    1.00  0.04978707   0.0
25%   3.25  0.25160736   0.0
50%   5.50  1.00000000   0.5
75%   7.75  5.05366896   1.0
100% 10.00 20.08553692   1.0
```

- a : beta 및 logic에 대한 출력값이 여러 개일 경우 행렬로 출력을 만듦

sweep 함수

- 개요
- 계산 보기 1
- 계산 보기 2

개요

적용

apply, lapply 등은 함수를 적용하는데 반해 sweep는 **연산자를 적용하는 함수**

사용 함수

```
sweep(x, MARGIN, STATS, FUN = "-",  ...)
```

- 행렬, 배열 또는 데이터 프레임이 자료 x에 대해서, 각 MARGIN(1=행, 2=열 등등)에 대해서 STATS에 주어진 값을 **FUN에 주어진 연산**을 함
- 연산의 기본값은 **뺄셈("-")**임

계산 보기 1

sweep 함수로 계산하기

자료

행렬인 원 자료에 대해서 첫 번째 행은 1, 두 번째 행은 2, 등등을 합해주는 경우를 생각해보자.

```
> x <- matrix(seq(1,25), byrow=T, ncol=5)
> x
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
[3,]   11   12   13   14   15
[4,]   16   17   18   19   20
[5,]   21   22   23   24   25
```

* 원자료 x가 얻어짐

계산 보기 1

sweep 함수로 계산하기

각 행에 1,2,..,5를 더해주기

```
> sweep (x, 1, STATS=seq(1,5), FUN="+")
     [,1] [,2] [,3] [,4] [,5]
[1,]    2    3    4    5    6
[2,]    8    9   10   11   12
[3,]   14   15   16   17   18
[4,]   20   21   22   23   24
[5,]   26   27   28   29   30
```

계산 보기 2

sweep 함수로 계산하기

자료

앞에서 본 학생의 성적 자료 데이터 프레임 score의 각 과목별 점수에서 각 과목별 평균으로부터의 편차를 계산해보자. (편차 = 자료값 - 평균)

※ 편차값이 양수이면서 큰 값이면 평균보다 우수한 점수, 편차가 음수로 큰 값이면 평균에 많이 미달되는 점수이다.

```
> colmean <- apply (score[,-1], 2 , mean) # score의 열별 평균
> sweep (score[,-1], 2,colmean, "-") # 평균으로 부터의 편차
```

	국어	영어	수학	과학	사회
1	-0.52	11.64	-5.74	5.14	4.52
2	-4.52	-2.36	16.26	-6.86	9.52
3	6.48	-22.36	10.26	-7.86	-17.48
4	-12.52	-10.36	-16.74	4.14	20.52
5	8.48	-6.36	3.26	12.14	-5.48

계산 보기 2

sweep 함수로 계산하기

계산의 결과 검산하기

```
> score[,2] - mean(score[,2])
[1]   -0.52   -4.52   6.48   -12.52   8.48   -8.52 ...
```

outer 함수

- 개요
- 연산자의 사용

개요

적용

벡터 x, y에 대해서 x의 i번째 원소와 y의 j번째 원소에 대해서 함수를 적용하고,
그 결과를 행렬 z[i,j]로 반환하는 함수

사용 함수

```
outer(x, y, FUNC="*", ...)
```

- FUN은 사용자 지정함수 설정 가능
- FUN의 기본값은 곱

연산자의 사용

연산자

```
> outer(seq(1,2), seq(1,3))
```

```
seq(1,2) %o% seq(1,3)
```

➢ 두 경우 모두 같은 결과

결과

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

연산자의 사용

연산자 또는 함수를 FUN에 설정한 경우

```
> outer(seq(1,3), seq(1,3), FUN="+")
     [,1] [,2] [,3]
[1,]    2    3    4
[2,]    3    4    5
[3,]    4    5    6
> outer(month.abb, 2011:2016, FUN = "paste")
      [,1]          [,2]          [,3]          [,4]          [,5]          [,6]
 [1,] "Jan 2011"    "Jan 2012"    "Jan 2013"    "Jan 2014"    "Jan 2015"    "Jan 2016"
 [2,] "Feb 2011"    "Feb 2012"    "Feb 2013"    "Feb 2014"    "Feb 2015"    "Feb 2016"
 [3,] "Mar 2011"    "Mar 2012"    "Mar 2013"    "Mar 2014"    "Mar 2015"    "Mar 2016"
 [4,] "Apr 2011"    "Apr 2012"    "Apr 2013"    "Apr 2014"    "Apr 2015"    "Apr 2016"
 [5,] "May 2011"    "May 2012"    "May 2013"    "May 2014"    "May 2015"    "May 2016"
 [6,] "Jun 2011"    "Jun 2012"    "Jun 2013"    "Jun 2014"    "Jun 2015"    "Jun 2016"
 [7,] "Jul 2011"    "Jul 2012"    "Jul 2013"    "Jul 2014"    "Jul 2015"    "Jul 2016"
 [8,] "Aug 2011"    "Aug 2012"    "Aug 2013"    "Aug 2014"    "Aug 2015"    "Aug 2016"
 [9,] "Sep 2011"    "Sep 2012"    "Sep 2013"    "Sep 2014"    "Sep 2015"    "Sep 2016"
[10,] "Oct 2011"    "Oct 2012"    "Oct 2013"    "Oct 2014"    "Oct 2015"    "Oct 2016"
[11,] "Nov 2011"    "Nov 2012"    "Nov 2013"    "Nov 2014"    "Nov 2015"    "Nov 2016"
[12,] "Dec 2011"    "Dec 2012"    "Dec 2013"    "Dec 2014"    "Dec 2015"    "Dec 2016"
```