R 프로그래밍 기초다지기

4강 - 리스트 완벽 마스터

슬기로운통계생활

Issac Lee

List를 배워보자.

리스트(List)이란 무엇일까?

언제 사용할까?

- 벡터의 경우 구성원들이 모두 같은 타입이여야 함.
- 각기 다른 타입의 구성원을 가진 객체를 만들 수는 없을까?

```
as.vector(c(1, "test"))
```

```
## [1] "1"    "test"
```

```
as.vector(c(1, TRUE))
```

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

```
as.vector(c(FALSE, TRUE))
```

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

리스트 (List) 만들기

장점

- 여러개의 다른 객체들 (Objects)을 모아놓을 수 있음

선언방법

- `list()` 함수를 사용하여 선언
- 각 구성원의 태그(tag)과 내용을 같이 설정해줌.

```
mylist <- list(name = "issac",
               id = 30096,
               order = c(1, 2))
mylist
```

```
## $name
## [1] "issac"
##
## $id
## [1] 30096
##
## $order
## [1] 1 2
```

벡터 vs. 리스트

공통점

- 작은 구성원들을 모아놓은 객체
- 리스트도 벡터의 한 종류

차이점

- `atomic vector`
  - 작은 구성원들로 쪼갤 수 없기 때문
- `recursive vector`
  - 작은 구성원들로 쪼개짐. (접근 가능)

```
mylist$name
```

```
## [1] "issac"
```

```
mylist$id
```

```
## [1] 30096
```

```
mylist$order
```

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

태그 (tag) 없이 선언하기

태크 사용

```
mylist
```

```
## $name
## [1] "issac"
##
## $id
## [1] 30096
##
## $order
## [1] 1 2
```

- 태그 사용하는 것을 권장함.

태크 미사용

```
mylist2 <- list("issac",
                30096,
                c(1, 2))
mylist2
```

```
## [[1]]
## [1] "issac"
##
## [[2]]
## [1] 30096
##
## [[3]]
## [1] 1 2
```

리스트 인덱싱(indexing)

구성원소 접근하기

- `$` 기호를 이용하여 접근

`[[]]` vs. `[]`

- 구성원소의 접근을 위해서 `[[]]`을 이용
- 원래 리스트의 부분을 잡아내기 위해서는 `[]`을 이용

```
mylist$name
```

```
## [1] "issac"
```

```
mylist[["name"]]
```

```
## [1] "issac"
```

```
mylist["name"]
```

```
## $name
## [1] "issac"
```

리스트 인덱싱(indexing)

숫자를 사용한 접근

- 상황에 따라서 편리한 접근 방법을 사용

외우기!

- `[[]]`의 결과는 구성원소 그 자체
- `[]`의 결과는 리스트

```
mylist[[1]]
```

```
## [1] "issac"
```

```
mylist[1]
```

```
## $name
## [1] "issac"
```

```
mylist2[1]
```

```
## [[1]]
## [1] "issac"
```

구성원소 추가/삭제/변경

변경 및 추가

- `$` 기호를 사용함.

```
mylist$id <- 202124
mylist
```

```
## $name
## [1] "issac"
##
## $id
## [1] 202124
##
## $order
## [1] 1 2
```

```
mylist$add <- "new element"
mylist
```

```
## $name
## [1] "issac"
##
## $id
## [1] 202124
##
## $order
## [1] 1 2
##
## $add
## [1] "new element"
```

구성원소 추가/삭제/변경

`NULL`을 사용한 삭제

```
mylist$add <- NULL
mylist
```

```
## $name
## [1] "issac"
##
## $id
## [1] 202124
##
## $order
## [1] 1 2
```

한번에 여러개 변경

```
mylist[1:2] <- list("jelly",
                    203149)
mylist
```

```
## $name
## [1] "jelly"
##
## $id
## [1] 203149
##
## $order
## [1] 1 2
```

리스트 안에 리스트 있다.

재귀 (recursive) 리스트

```
mylist$new <- list("hello",
                   c(1, 3, 2))
mylist
```

- 리스트 안에 리스트 있다.
- 어떻게 접근할까?

```
## $name
## [1] "jelly"
##
## $id
## [1] 203149
##
## $order
## [1] 1 2
##
## $new
## $new[[1]]
## [1] "hello"
##
## $new[[2]]
## [1] 1 3 2
```

재귀 리스트 접근방법

해석 연습

- 옆의 코드를 보고 결과를 이해하자!
- `mylist[1][2]`의 결과는?
  - 왜 안될까?

```
mylist$new[1]
```

```
## [[1]]
## [1] "hello"
```

```
mylist$new[[1]]
```

```
## [1] "hello"
```

```
mylist$new[[2]][2]
```

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

리스트와 언리스트(Unlist)

`unlist()`를 사용한 변환

- 문법 `unlist(x, recursive = TRUE, use.names = TRUE)`

- `recursive` 옵션
  - 안에 들어있는 리스트들도 `unlist()`를 적용할 것인가?
  - `unlist(mylist, recursive = FALSE)` 확인

```
unlist(mylist)
```

```
##     name       id   order1
##  "jelly" "203149"      "1"
```

리스트 합치기 (Concatenating)

```
c(list(1, "2"),
  list(5, c(1, 3)))
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] "2"
##
## [[3]]
## [1] 5
##
## [[4]]
## [1] 1 3
```

`recursive` 옵션을 사용한 `unlist`

```
c(list(1, "2"),
  list(5, c(1, 3)),
  recursive = TRUE)
```

```
## [1] "1" "2" "5" "1" "3"
```

리스트에 함수 적용하기

벡터에 함수 적용

- 벡터가 주어졌을때 함수 적용이 너무 쉬웠음.

```
sum(1:10)
```

```
## [1] 55
```

- 벡터가 많다면 함수를 일일이 적용해야 할까?

```
sum(1:10)
```

```
## [1] 55
```

```
sum(112:120)
```

```
## [1] 1044
```

```
sum(20:40)
```

```
## [1] 630
```

```
#...
```

`lapply()`와 `sapply()` 함수

리스트의 각 원소에 동일한 함수를 적용

- `lapply(list, function)`
- `l(ist)apply()` 함수는 리스트 각 원소에 접근해서 같은 함수를 적용시켜 줌.
    - 결과는 리스트로 나옴.
- `simplied (l)apply()` 함수는 결과를 단순화시켜서 벡터형식으로 뱉어냄.

```
lapply(list(1:2, 1:5), sum)
```

```
## [[1]]
## [1] 3
##
## [[2]]
## [1] 15
```

```
sapply(list(1:2, 1:5), sum)
```

```
## [1]  3 15
```

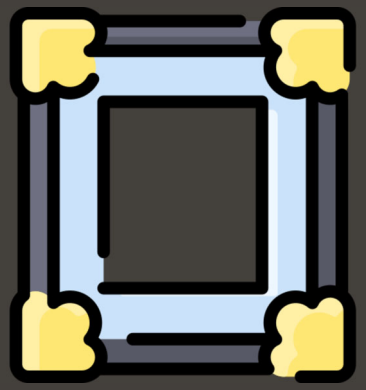
연습문제

- 각 벡터들의 최대값의 위치는?

```
set.seed(2021)
mylist3 <- replicate(100, runif(sample(10:50, 1)))
str(mylist3)
```

```
## List of 100
##  $ : num [1:16] 0.784 0.71 0.382 0.636 0.701 ...
##  $ : num [1:32] 0.958 0.546 0.14 0.955 0.392 ...
##  $ : num [1:24] 0.0728 0.5312 0.3617 0.1874 0.4627 ...
##  $ : num [1:40] 0.289 0.693 0.995 0.546 0.409 ...
##  $ : num [1:32] 0.0661 0.0462 0.5064 0.2592 0.4265 ...
##  $ : num [1:16] 0.0361 0.1589 0.6024 0.0819 0.6896 ...
##  $ : num [1:10] 0.4588 0.1075 0.7647 0.7033 0.0264 ...
##  $ : num [1:22] 0.237 0.385 0.616 0.479 0.336 ...
##  $ : num [1:22] 0.443 0.267 0.313 0.405 0.116 ...
```

다음시간



데이터 프레임

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?