

함수의 출력 방법

- 📖 print, cat, return 함수 사용
- 📖 list 함수 사용
- 📖 요약 정리

Lorem ipsum dolor sit amet, ius an molestie facilisi erroribus, mutat nalerum delectus ei vis. Has ornatus conclusionemque id, an videri molestatis sit. In etqui praesent sit. An vel agan porro comprehensan, ad ludus constituto nea, et ius utroque scaevola assuaverit.

Vis cu nodus nulla feugait, oratio facilisi ex usu, eili vitae sea te. Ea fabulas accusamus dissentias sea, facete tacinales definitiones et per. Nihil dicant mediocrem pro eu, no mei nostro sensibus platonem. Qui id sunno perpetua neglegentur. Vel ipsum novum copiosae ut. Quo et liber detracto probatus. Nam augue scribentur an. Sea oporteat percipitur incidere at. Qui viris nemore an.



print, cat, return 함수 사용



print 함수 사용

- + 단순히 결과를 화면에 인쇄하려는 경우 사용

```
myfcn1 <- function() {  
  x <- seq(1,10)  
  print(sum(x))  
}
```

- + 계산 결과

```
> myfcn1()  
[1] 55
```



print, cat, return 함수 사용



cat 함수 사용

- + 출력에 적절한 문자열 등을 함께 사용하려는 경우 사용

```
myfcn2 <- function() {  
  x <- seq(1,10)  
  cat("sum of x: ", sum(x), "\n")  
}
```

- + 계산 결과

```
> myfcn2()  
sum of x: 55
```

* 문자열을 추가로 설정할 수 있음 → 숫자 55가 무엇인지 설명이 추가됨

[참고] \n 은 줄 바꾸기를 인쇄하므로 다음 입출력이 새 줄에서 시작하도록 함



print, cat, return 함수 사용



return 함수

- + 함수 값의 반환은 기본적으로 return 함수를 사용하며, 한 개만 return할 수 있음

```
myfcn3 <- function() {  
  x <- seq(1,10)  
  return(sum(x))  
}
```

- + 계산 결과

```
> myfcn3()  
[1] 55
```

* 여러 개의 출력은 list 함수를 사용하는 것을 권장



print, cat, return 함수 사용



세 방법의 비교



함수의 결과값에 1000을 합하는 계산을 해본 결과

Print 함수

```
> myfcn1()+1000
```

```
[1] 55
```

```
[1] 1055
```

Cat 함수

```
> myfcn2()+1000
```

```
sum of x: 55
```

```
numeric(0)
```

return 함수

```
> myfcn3()+1000
```

```
[1] 1055
```



print, cat, return 함수 사용



세 방법의 비교 결과

- + print와 return 함수는 한 개체 매개변수로 사용 가능하므로 **둘 이상의 개체를 반환하기에 적절하지 않음**
 - 둘 이상의 개체를 print나 return 함수로 반환하려면 c 또는 list 등의 함수를 사용하여 반환할 값을 하나로 묶어야 함
- + print와 return 함수를 사용하여 결과를 반환한 경우, 함수의 반환값을 다른 연산에 직접 사용 가능하지만 **cat 함수는 결과를 다른 연산에 사용하지 못함**
- + print 함수는 반환값을 무조건 인쇄하고 반환값을 사용한 추가 연산을 함
 - 중간 출력이 필요 없는 경우 이 함수는 권장되지 않음
- + cat 함수는 함수의 계산 결과가 최종 결과이며 반환값으로 추가 계산을 하지 않는 경우를 제외하면 함수값 반환의 방법으로 권장되지 않음



list 함수 사용



개요

- + 함수의 출력을 추후 출력을 사용하여 다른 함수들을 사용하려는 경우 사용
- + 각 개체 obj에 대해서 출력의 이름 name 설정 예제

```
list(name1=obj1, name2=obj2, ...)
```

- + 여러 개의 값에 대한 반환이 가능하며 하나의 값만 반환받으려면 \$를 사용하거나 []을 사용하여 해당 인덱스를 지정하면 됨
- + list 함수의 장점 중의 하나는 각각의 출력에 사용자가 이름을 설정할 수 있음



list 함수 사용



예제



자료

xx를 매개변수로 받아서 xx의 합, xx의 제곱 합 두 개를 sum1, sum2로 list 함수로 반환해보자.

```
myfcn4 <- function(xx) {  
  sum1 <- sum(xx)  
  sum2 <- sum(xx^2)  
  list(sum1=sum1, sum2=sum2)  
}
```



평균과 실험계획 또는 회귀분석에서 사용하는 SST($SST = \sum y_i^2 - n\bar{y}^2$) 계산

위 계산식에서

1. sum1/n을 계산하고 이를 mx함
2. $SST = \text{sum2} - n * \text{mx}^2$ 을 계산함

* n은 자료의 수

* sum1과 sum2 : 함수에서 얻은 반환값



list 함수 사용



\$로 개별 개체 불러오기

+ 매개변수 x로 계산한 함수 myfcn4의 반환값 중에서

sum1만 얻으려는 경우

```
> myfcn4(x)$sum1
```

sum2만 얻으려는 경우

```
> myfcn4(x)$sum2
```

+ 위의 함수 반환값을 \$를 사용하여 따로 불러서 평균과 SST를 계산

```
> mx <- myfcn4(seq(1,10))$sum1 / 10  
> SST <- myfcn4(seq(1,10))$sum2 - 10*mx^2  
> SST  
[1] 82.5
```



list 함수 사용



\$로 개별 개체 불러오기



참고사항

- myfcn4를 두 번 호출하여 계산을 두 번 하는 경우처럼 단순 계산을 많이 반복하지 않는 경우에는 문제가 없으나 복잡한 계산을 반복하는 경우에는 CPU 시간이 늘어남
- 함수 호출을 한 번만 하도록 다음과 같이 사용하는 것이 권장됨

```
> result <- myfcn4(seq(1,10))  
> mx <- result$sum1 / 10  
> SST <- result$sum2 - 10*mx^2  
> SST  
[1] 82.5
```





list 함수 사용



첨자로 특정 반환값 호출하기



list 함수로 함수의 계산 결과를 반환한 경우 첫 번째 값은 첨자 [1], 두 번째 값은 첨자 [2] 등을 기본으로 갖게 되므로 인덱스를 사용하여 개별 값을 반환 받을 수도 있음

첫 번째 값은 첨자 [1]

```
> myfcn4(seq(1,10))[1]  
$sum1  
[1] 55
```

두 번째 값은 첨자 [2]

```
> myfcn4(seq(1,10))[2]  
$sum2  
[1] 385
```



에러 메시지가 발생하는 경우

```
> mx <- myfcn4(seq(1,10))[1] / 10
```




인덱스를 사용하여 값을 반환 받는 경우 그 값을 사용한 추가 연산이 곤란함

➔ list 함수를 사용하여 여러개의 값은 함수 계산의 결과로 반환 받을 때는 \$의 사용이 권장됨



함수를 이용한 새로운 연산자 만들기

 함수를 이용한 새로운 연산자의 정의
및 계산



함수를 이용한 새로운 연산자의 정의 및 계산



연산자의 시작과 끝은 반드시 %



정의할 때는 따옴표를 사용하여 연산자를 정의



이 연산자를 사용할 때는 따옴표 없이 사용



연산자 %abinb%를 정의하고 이 함수는 $a*2+b$ 를 계산함

```
> "%abinb%" <- function(a,b){ return (a*2 +b) }  
> 1 %abinb% 1  
[1] 3
```

재귀적 호출(recursive calls)

- 📖 재귀적 호출의 개념
- 📖 재귀적 호출의 사용

Lorem ipsum dolor sit amet, ius an molestie facilisi erroribus, mutat nalerum delectus ei vis. Has ornatus conclusionemque id, an vide molestatis sit. In etqui praesent sit. An vel agan porro comprehensan, ad ludus constituto nea, et ius utroque scaevola assuaverit.

Vis cu nodus nulla feugait, oratio facilisi in usu, eilit vitae sea te. Ea fabulas accusamus dissentias sea, facete tacinates definitiones et per. Nihil dicant mediocrem pro eu, no mei nostro sensibus platonem. Qui id sunno perpetus neglegentur. Vel ipsun novum copiosae ut. Quo et liber detracto probatus. Nam augue scriben- tur an. Sea oporteat percipitur incidereat ab. Qui viris nemore an.



재귀적 호출의 개념



재귀적 호출이란 어떤 함수가 자기 자신을 호출하는 것



$n!$ 의 값 계산식 : $n! = n \times (n-1) \times \cdots \times 1$



사용 함수

```
nfactorial1 <- function(nn) {  
  result <- 1  
  for (i in 1:nn) {  
    result <- result*i  
  }  
  return(result)  
}
```



계산 결과

```
> nfactorial1(5)  
[1] 120
```





재귀적 호출의 사용



$n! = n \times (n-1)!$ 이므로 factorial을 계산하는
왼쪽 항에 오른쪽 항에도 같은 factorial 계산이 있음

```
nfactorial2 <- function(nn) {  
  if (nn == 1) result <- 1  
  else result <- nn * nfactorial2(nn-1)  
  return(result)  
}
```

+ 계산 결과

```
> nfactorial2(10)  
[1] 3628800  
> nfactorial2(3)  
[1] 6  
> nfactorial2(4)  
[1] 24  
> nfactorial2(5)  
[1] 120
```



재귀적 호출의 사용



재귀적 호출 적용하기

- + n 이 1이면 결과는 1 ($\text{factorial}(1)=1$)
- + n 이 2이면 결과는 $2 * \text{factorial}(1)=2 * 1$
- + n 이 3이면 결과는 $3 * \text{factorial}(2)$ 인데 $\text{factorial}(2)$ 는 다시 $2 * \text{factorial}(1)$ 이므로 결과는 $3 * 2 * 1$



참고사항

- + R-언어의 내장함수 `factorial`이 $n!$ 을 계산해줌

