분석프로그래밍 I.09
제어와 함수 2

국민대학교 경영학부 빅데이터경영통계학 전공

2018. 5. 14(월)

1 R Markdown

2 함수

- 스크립트와 같이 반복해서 활용하게 될 코드를 모아 놓은 것.
- **매번 동일하게 실행되는 부분**과 **달라지는 부분**으로 구분해 볼 수 있다.
- 예) 다음은 1에서 5까지의 합, 1에서 10까지의 합, 1에서 20까지의 합을 구하고 있다.

```
s=0
for {i in 1:5} {
  s = s + i
  }
print(s)

s=0
for (i in 1:10) {
  s = s + i
  }
print(s)

s=0
for (i in 1:20) {
  s = s + i
  }
print(s)
```

Listing 1: 1부터 n까지 더하기

- 여기서 공통되는 부분을 함수로 나타내는 방법은 다음과 같이 진행할 수 있다.

1) 공통되는 부분을 확인한 후, 달라지는 부분을 변수로 치환한다.

2) 전체를 `{ }`로 감싼다.

3) 반환값을 명시한다: `return( )`

4) 함수의 이름과 인자를 명시한다: `= function(a, b, ...)`

```
s=0
for {i in 1:n} {
  s = s + i
  }
print(s)
```

Listing 2: 달라지는 부분을 변수로 나타낸다.

```
{
  s=0
  for {i in 1:n} {
    s = s + i
    }
  print(s)
}
```

Listing 3: 전체를 `{ }`로 감싼다.

```
{
  s=0
  for {i in 1:n} {
    s = s + i
    }
  print(s)
  return(s)
}
```

Listing 4: 반환값을 명시한다.

```
sumToN = function(n) {
  s=0
  for {i in 1:n} {
    s = s + i
    }
  print(s)
  return(s)
}
```

Listing 5: 함수의 이름과 인자를 명시한다.

3 함수의 인자

- 인자가 여럿일 때 구분하는 방법은 1) 이름으로, 2) 위치로, 3) 부분 매칭(partial matching)이 있다.

```
sumAToB = function(startNumber, endNumber) {
  s=0
  for {i in startNumber:endNumber} {
    s = s + i
    }
  print(s)
  return(s)
}
```

Listing 6: `startNumber`에서 `endNumber`까지 더하는 함수

1) 이름으로 : `sumAToB(startNumber=1, endNumber=5)`, `sumAToB(endNumber=5, startNumber=1)`

2) 위치로 : `sumAToB(1, 5)`

3) 부분 매칭으로 : `sumAToB(startNum = 1, end=5)`

- 인자는 기본값(default value)을 설정해 줄 수 있고, 기본값이 주어진 인자는 함수를 부를 때 생략할 수 있다.

```
sumAToB = function(startNumber=1, endNumber=10) {
  s=0
  for {i in startNumber:endNumber} {
    s = s + i
    }
  print(s)
  return(s)
}
```

Listing 7: 기본값이 주어진 인자들

- 함수는 ...(dot-dot-dot)을 활용하여 불특정 다수의 인자를 받아들일 수 있다. 이때에는 부분 매칭이 불가능하다. 보통 **그 밖의** 인자들을 다른 함수로 넘길 때 사용된다.

```
sqPlot = function(x, y, ...) {
  plot(x^2, y, ...)
}
```

Listing 8: ... 활용의 예

- 만약 함수가 제대로 작동하기 위해 인자에 특별한 조건이 있다면 stopifnot()을 사용할 수 있다.

```
boxcox = function(x, lambda=1)
{
  stopifnot(length(lambda)==1)
  if (lambda != 0) {
    return((x^lambda-1)/lambda)
  } else {
    log(x)
  }
}
```

Listing 9: stopifnot 활용의 예

4 함수와 인자의 클래스

- Generic function: 인자의 클래스(`class`)에 따라 함수의 행동이 달라지는 함수이다. `methods( )`를 통해 클래스에 따른 함수를 확인할 수 있다.

```
1 methods(print)
2 methods('print')
```

Listing 10: 제너릭 함수인 `print`에 대해 `methods( )`

5 그 밖의 몇 가지

- R은 기본적으로 Call by value이다. 함수의 인자는 인자의 주소가 아니라 값이 함수로 전달된다.
- 함수 내에서 사용된 변수는 함수가 끝나는 순간 사라진다.
- 함수 밖에서 할당된 변수는 참조할 수 있지만 '`=`' 또는 '`<-`'로 수정할 수 없다. 따라서 R의 함수 내 변수는 흔히 말하는 지역 변수(local variables)라고 할 수 있다.
- 만약 함수 밖에서 할당된 변수에 새로운 값을 할당하려면 연산자 `<<-`를 사용한다.

6 디버깅(Debugging)

R studio에는 함수 내부를 디버깅할 수 있는 도구가 마련되어 있다. 특히 break-point를 지정하는 방법을 숙지하자. 에러가 발생되었을 때, *Show Traceback*, *Rerun with Debug* 등을 사용할 수 있다.