Pipe and Function

박찬영

2024-08-31

tidyverse와 magrittr library를 사용합니다. 파이프 연산자를 쓰는데는 magrittr이 좋다. (tidyverse에 있음)

파이프 연산자

파이프 연산자는 연속적으로 여러단계의 함수를 쓸 때 유리하다

파이프를 쓸 때는 동사에 집중

허나 파이프 단계가 10단계 이상 길어지면 중간 객체를 만들어놓자 선형적인 처리에 유용하므로 비선형적 처리는 ㄴㄴ

함수

함수는 같은 작업의 뭉침정도라고 생각할 수 있다

```
#함수 정의
change = function(x,y) {
    temp=x
    x=y
    y=temp
    return(c(x,y))
}

change(1,2)
```

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

```
#다른 언어의 함수와 마찬가지로 직접 변수의 데이터를 바꾸지 않는다
#함수의 이름은 기본적으로 동사

#생활 꿀팁, if문에는 부울형 벡터를 넣으면 안된다 (NA도 넣으면 안됨..)

mean_ci = function(x, conf = 0.95) {
```

```
  se = sd(x) / sqrt(length(x))
  alpha = 1 - conf
  mean(x) + se * qnorm(c(alpha / 2, 1 - alpha / 2))
}

#함수의 인수에는 기본값을 만들어놓을 수 있다
x = runif(100)
mean_ci(x)
```

```
## [1] 0.4823462 0.5991840
```

```
mean_ci(x, conf=0.99) #기본값을 바꿀 때는 무조건 풀인수네임을 써줘야한다
```

```
## [1] 0.4639896 0.6175405
```

```
#mean_ci(x, 0.99) no

commas = function(...) str_c(..., collapse = ", ")
#인수에 ...으로 하면 받는 데이터의 수가 정해지지 않은 것이다
commas(letters[1:10])
```

```
## [1] "a, b, c, d, e, f, g, h, i, j"
```

```
f = function(x) {
    x+y
}

#y는 함수안에서 정의되지 않았다
#f(10) #에러가 뜸
#y=100
#f(10) #함수 밖에서 정의된 y가 함수 안에 영향을 준다

`+` <- function(x, y) {  # override `+`
  if (runif(1) < 0.1) {
    sum(x, y)
  } else {
    sum(x, y) * 1.1
  }
} #+을 새롭게 정의
table(replicate(1000, 1 + 2)) #새로 정의된 덧셈에 대한 연산이라 3과 3.3이 다나옴
```

```
##
```

```
##   3 3.3
## 103 897
```

```
rm(`+`) #새로 정의된 덧셈 제거
```

파이프와 함수

파이프 연산자를 쓸 수 있는 함수를 만들어보자. 파이프 연산자를 쓸 수 있는 함수는 무조건 데이터프레임을 반환해야한다. 1. 첫번째 인수로 데이터를 받고, 변환하여 반환해야한다. 2. 데이터를 변환하지는 않고, 데이터에 어떠한 작업을 취한다.

이런 두가지 경우의 수가 존재한다.

```
show_missing = function(df) {
    n= sum(is.na(df))
    cat("Missing values: ",n,"\n", sep="")

    invisible(df) #보여주지않는 반환이다
}

show_missing(mtcars) #작동잘됨
```

```
## Missing values: 0
```

```
class(show_missing(mtcars)) #이건 df죠?
```

```
## Missing values: 0
```

```
## [1] "data.frame"
```

```
dim(show_missing(mtcars)) #안보이게 반환됨
```

```
## Missing values: 0
```

```
## [1] 32 11
```

```
mtcars %>%
    show_missing() %>%
    mutate(mpg = ifelse(mpg<20, NA, mpg)) %>%
    show_missing()
```

```
## Missing values: 0
## Missing values: 18
```

```
#show_missing은 데이터를 변환하진 않으며 데이터의 정보를 출력하고 원본 데이터를 다시 반환해준다
```