Functions - r4ds

Tomoya Fukumoto

2019-07-26

関数

神のお言葉

データサイエンティストとしてレベルアップする最高の方法は関数を書くこと

関数の利点(コピペに対する)

1. ある処理の塊に名前を付けて管理できる。
   - コードが理解しやすくなる
2. 変更があったときに一箇所だけ修正すればよい
   - 生産性 UP!
3. コピペしたときのミスの可能性を減らせる
   - 不具合の減少

19.2 When you should you write a function?

どういうときに関数を書くのか？

A. コピペの回数が２回を上回るとき

Don’t Repeat Yourself (DRY) principle

関数を書くべき例

```
df <- tibble::tibble(
                a = rnorm(10),
                b = rnorm(10),
                c = rnorm(10),
                d = rnorm(10))

df$a <- (df$a - min(df$a, na.rm = TRUE)) /
  (max(df$a, na.rm = TRUE) - min(df$a, na.rm = TRUE))
df$b <- (df$b - min(df$b, na.rm = TRUE)) /
  (max(df$b, na.rm = TRUE) - min(df$a, na.rm = TRUE))
df$c <- (df$c - min(df$c, na.rm = TRUE)) /
  (max(df$c, na.rm = TRUE) - min(df$c, na.rm = TRUE))
df$d <- (df$d - min(df$d, na.rm = TRUE)) /
  (max(df$d, na.rm = TRUE) - min(df$d, na.rm = TRUE))
```

関数を書くための step1 コードを分析する

次の処理の入力は？

```
(df$a - min(df$a, na.rm = TRUE)) /
    (max(df$a, na.rm = TRUE) - min(df$a, na.rm = TRUE))
```

答え

```
x <- df$a
(x - min(x, na.rm = TRUE)) /
  (max(x, na.rm = TRUE) - min(x, na.rm = TRUE))
```

関数を作る

```
rescale01 <- function(x) {
  rng <- range(x, na.rm = TRUE)
  (x - rng[1]) / (rng[2] - rng[1])
}
```

方法

1. 関数の名前を決定する
   - `rescale01`
2. 入力、または引数を `function` の中に入れる
3. 関数の内容を `function(...)` の後に続く `{` のブロックで表現

関数を使ってコードを書き直す

```
df$a <- rescale01(df$a)
df$b <- rescale01(df$b)
df$c <- rescale01(df$c)
df$d <- rescale01(df$d)
```

すっきりした！

まだコピペが残ってるやん

⇒ 次の次の章 iteration を待て