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

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

19.3 Functions are for humans and computers

関数はコンピュータのためだけではなく人間のために書く

- 関数名
- コメント

関数名の決め方

- 短く、しかし処理の内容を明確に表現したい
  - トレードオフになることも多い
  - どちらか一方を選択するとすれば明確にする
    - RStudio のオートコンプリート機能
- 名詞ではなく動詞で
  - 引数は名詞
  - 動詞が“get”や“compute”ならその目的語（名詞）もあり
- よりよい名前が見つかったら変更することをためらうな
- 関数群を作るときは前半部を統一する
  - オートコンプリート

関数名の例

```
# Too short
f()

# Not a verb, or descriptive
my_awesome_function()

# Long, but clear
impute_missing()
collapse_years()
```

関数群の命名

```
# Good
input_select()
input_checkbox()
input_text()

# Not so good
select_input()
checkbox_input()
text_input()
```

命名規則

- `snake_case`
  - Hadley おすすめ
- `camelCase`

どっちでもいいけど、どちらかに統一するべき

コメント

でその行の # より右側はコメントアウト

```
# this is comment
a <- pi #this also comment
```

- コメントは「なぜ」その処理をするのかを書くべし
  - 「何を」「どのようにして」では無い
- なぜ
  - 中間変数を置いたのか？
  - ２つの関数に分けたのか？
  - 他の方法は試したけどうまくいかなかったのか？
- なぜこのようなコードになったのか？

コードセクション(Rstudio)

次のコードを使えばRStdudioでセクション区切りとみなされる

```
# Load data ----------------------------------

# Plot data ----------------------------------
```

nav

キーボード・ショートカット

Ctrl + Shift + R