



# ば<sup>ん</sup>ぬ先の テスト・アサートの導入

～再現性の保証対策として～



瓜生真也 (🐣@uribo)

---

**転** こんなことが**頻繁に**ありますね

- ① 以前と異なる値が出た
- ② 文字列型だと思ったら  
因子型になっていた
- ③ なぜ・どこでエラーに  
なったのかがわからない

# データ は 変わる

## 色々な条件・環境の違いがある

- OS, R本体・パッケージのバージョン

## 変化（異常）に気がつくことが大事

- 思い込みのまま解析を続けてしまったら？
- どこで変化があったか探すのは徒労
- 何が変わってしまったのかを把握したい

テスト、アサートを導入することで、  
データの変化に気付きやすくなる

# テスト と アサート

## 単体テスト (unit test) と呼ばれるもの

- 本来は、プログラムが期待された通りに動作するかを確認するための作業
- 記述されたテストパターンを試す  
(記述のないテスト結果は保証しない)

## アサート (assert…断言する)

- 値の状態を予測する値を記述
- 予想外の値が与えられた際にエラーを出力

**library**  
**(testthat)**

# expect\_\*(): オブジェクト と 返り値の 状態を記述



```
expect_equal(  
  object = dim(iris),  
  expected = c(150, 5))
```



```
expect_equal(  
  dim(iris),  
  c(120, 4))
```

**Error: dim(iris) not equal to c(120, 4).**  
**2/2 mismatches (average diff: 15.5)**  
**[1] 150 - 120 == 30**  
**[2] 5 - 4 == 1**

# 関数

# 対象

# 例

`expect_equal()`

平等性

```
expect_equal(letters[1:3], c("a", "b", "c"))  
expect_setequal(letters[1:3], c("a", "c", "b"))
```

`expect_gt()`

大小関係

```
expect_gt(pi, 3.1)  
expect_gte(pi, 3.14)  
expect_lt(pi, 3.2)
```

`expect_true()`

真偽値

```
expect_true(iris$Species[1] == "setosa")
```

`expect_length()`

長さ

```
expect_length(unique(iris$Species), 3)
```





实践





# **`data(iris)` の状態を記録した**

## **6つのテスト項目を用意**

- データのサイズ（行・列の数）
- 列の名前
- Species列に含まれる水準



# テスト対象のオブジェクトを `my_iris.csv`に変えて再度実行

- `my_iris` は `iris` と同じデータである想定
- `iris` に対していくつかの加工を行ったもの

Error: Test failed: 'Iris data statement'

\* `dim(my_iris)` not equal to `c(150, 5)`.

1/2 mismatches

[1] 120 - 150 == -30

\* `unique(my_iris$Species)` has length 4, not length 3.

\* `levels(my_iris$Species)` not set-equal to `c("setosa", "versicolor", "virginica")`.

Lengths differ: 4 is not 3

「正しく」失敗する

**library**  
**(assertr)**

パイプ処理  
(%>%)  
フレンドリー  
な  
アサート  
関数を提供

```
library(dplyr)
```


```
iris %>%
```

```
  tibble::as_tibble() %>%
```

```
  select(Sepal.Length, Species) %>%
```

```
  verify(列に含まれる名前である  
    has_all_names(  
      c("Sepal.Length", "Species")))) %>%
```

```
  filter(Sepal.Length >= 5.0) %>%
```

```
  assert(範囲に含まれる値である  
    within_bounds(5.0, Inf), Sepal.Length)
```



实践

# アサートがないと...

src/02-assertr\_demo.R



```
my_iris <- iris
```

```
my_iris %>%
```

```
  select(Sepal.Length, Species) %>%
```

```
  filter(Sepal.Length >= 5.0) %>%
```

```
  group_by(Species) %>%
```

```
  summarise(
```

```
    sl_mean = mean(Sepal.Length))
```

```
# A tibble: 3 x 2
  Species    sl_mean
  <fct>      <dbl>
1 setosa      5.23
2 versicolor  5.96
3 virginica   6.62
```

# アサートがないと...

src/02-assertr\_demo.R

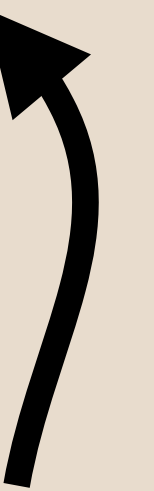


```
my_iris <-  
  read.csv(  
    here::here("data", "my_iris.csv"))  
  
my_iris %>%  
  select(Sepal.Length, Species) %>%  
  filter(Sepal.Length >= 5.0) %>%  
  group_by(Species) %>%  
  summarise(  
    sl_mean = mean(Sepal.Length))
```



# A tibble: 4 x 2  
Species sl\_mean  
<fct> <dbl>

1	setosa	5.28
2	versicolor	5.98
3	virginica	6.65
4	Virsicolor	6.2



これは望んだ結果?



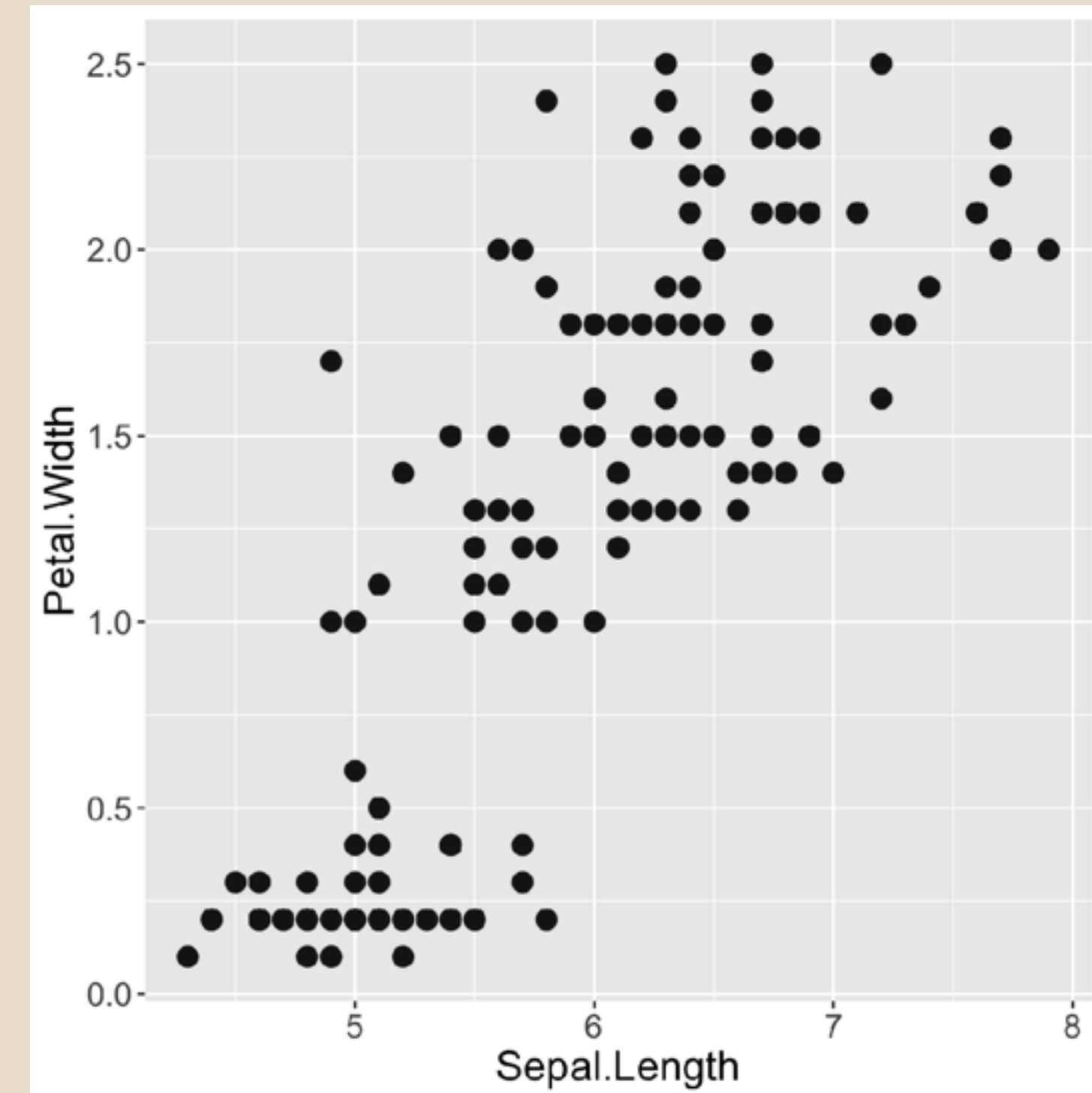
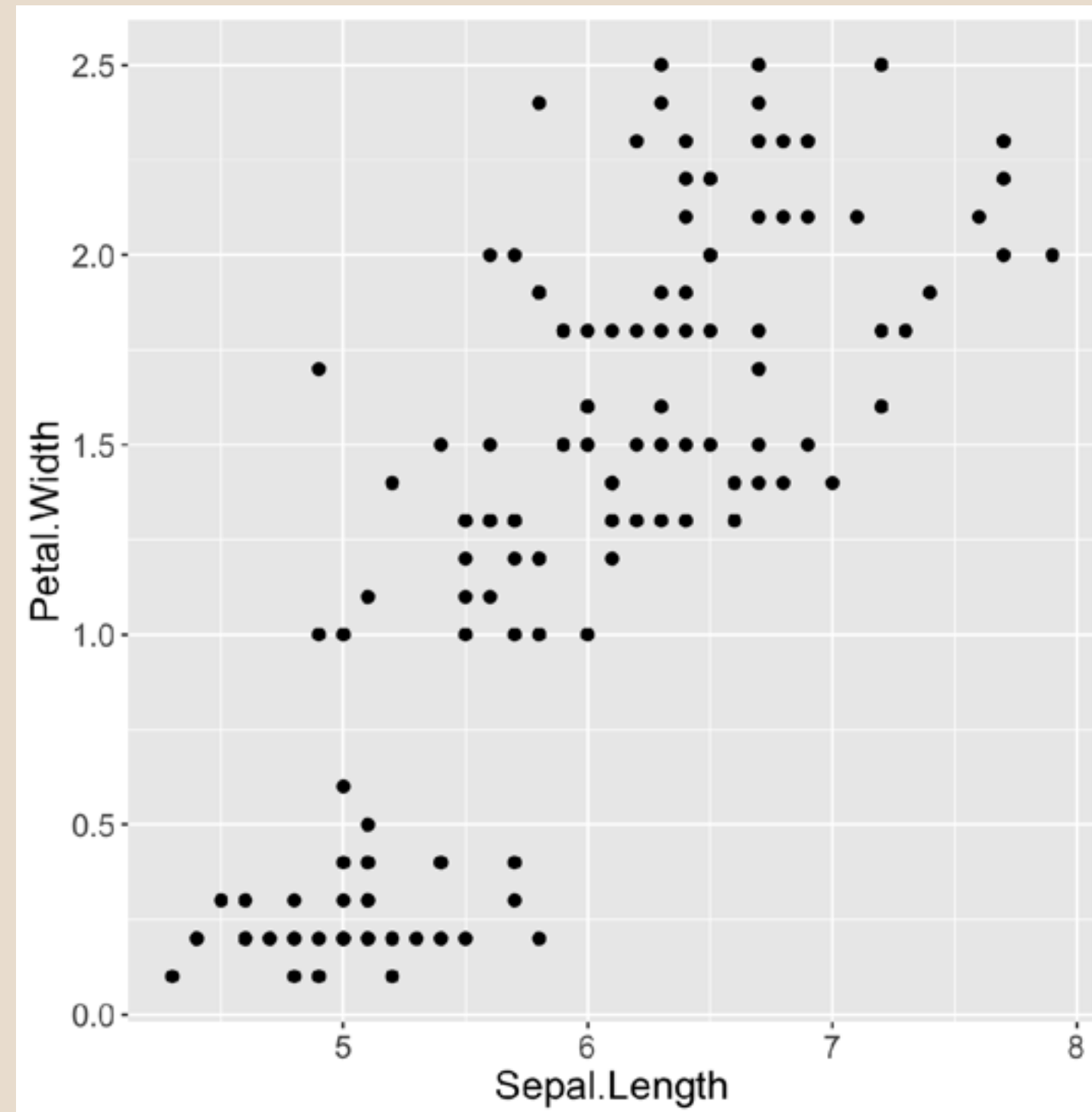
**library**  
**(vdiff)**

おまけ

その  
図は  
本当に  
同じ？

## 作図したファイルもテストしたい

- フォント、色、レイアウト etc.
- 変更を加えつつ、過去の図とも比較したい



ポイントの大きさと…? (探さないでください)

※パッケージ  
環境での  
利用を想定

- 1 作図を行う処理または関数を用意
- 2 `vdiffr::validate_cases()` を実行  
→ `tests/figs/` にSVGで出力される
- 3 `tests/testthat/` の中のファイルで  
1 を実行  
または `vdiffr::collect_cases()`,  
`vdiffr::diffrAddin()` でも可



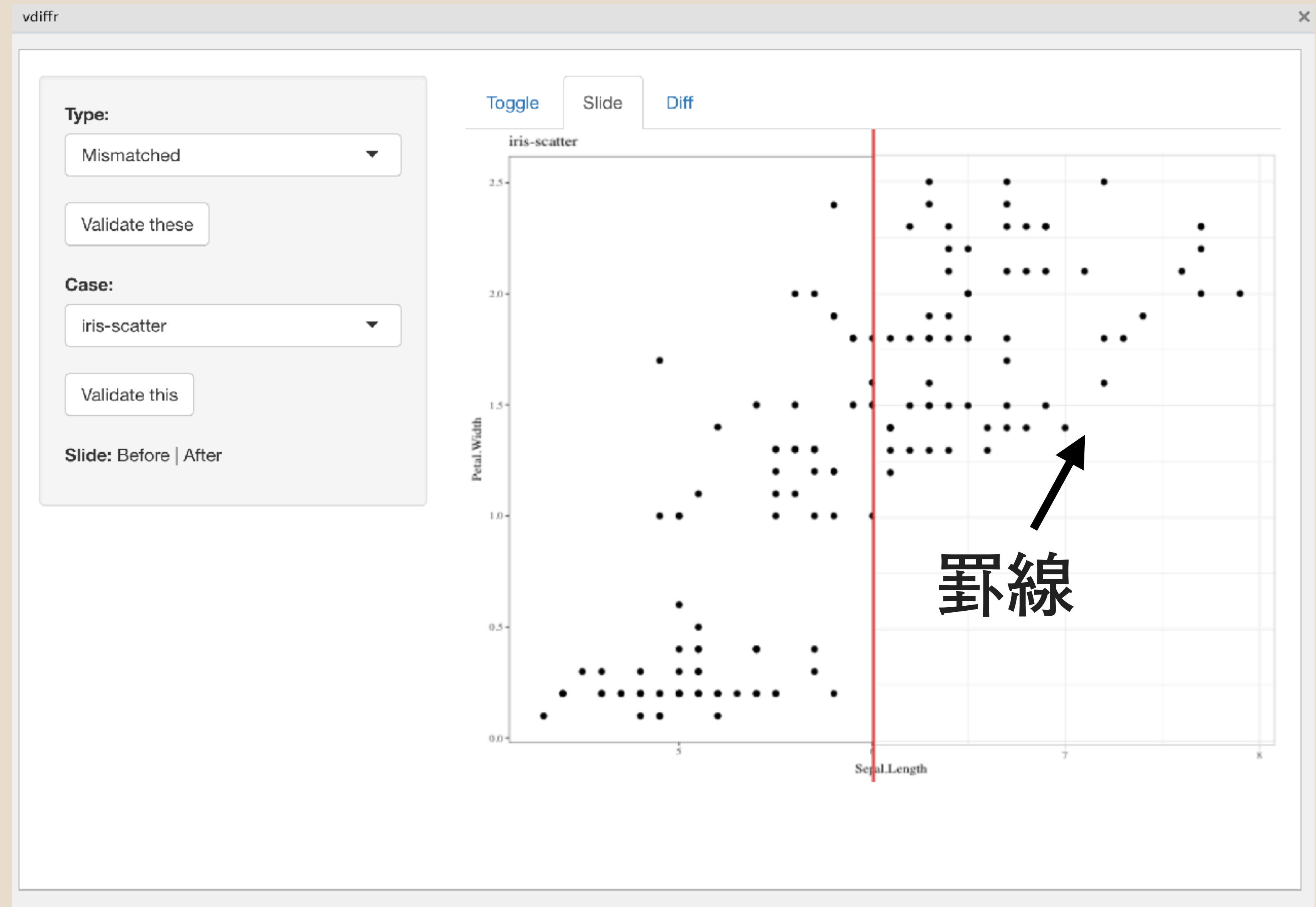
实践

# 作図処理のコードを修正

`src/03-plot.R`



前後の状態  
を比較



～最後に～

何を  
どこまで  
テスト  
するか

- テストを書くことで生産性は一次的に下がる
- 全てを網羅するのは困難
- データ数や欠損、列のデータ型などが大事？
- 正しいテストだけでなく失敗するケースの記述も重要

# 参加者の声から

- コメントによる値の記述はしている
- 目grepによる作業は避けたいのでGood
- テストに頼りすぎず、目視による確認も大事。

おわり