

Utilisation de map()

Les fonctions dérivées de `purr::map()` permet d'ajouter une colonne à un tableau avec `mutate()` avec une fonction non vectorielle.

Usage normal de `mutate()` avec une fonction vectorielle :

```
library("tidyverse")
# Tibble à une colonne: x.
tibble(x = 1:5) %>%
  # Pair ou impair ?
  mutate(parite = ifelse(x %% 2 == 0, "pair", "impair"))
```

```
# A tibble: 5 x 2
  x     parite
  <int> <chr>
1     1 impair
2     2 pair
3     3 impair
4     4 pair
5     5 impair
```

Si une fonction non vectorielle est nécessaire (ici : `if ()`), alors `map()` est capable de s'adapter :

```
tibble(x = 1:5) %>%
  mutate(
    parite = map_chr(
      .x,
      \((x) {if (x %% 2 == 0) "pair" else "impair"}))
```

```
# A tibble: 5 x 2
  x     parite
  <int> <chr>
1     1 impair
2     2 pair
3     3 impair
4     4 pair
5     5 impair
```

`map_chr()` renvoie un vecteur de caractères (`map()` renvoie toujours une liste). Son premier argument est le vecteur d'entrée, et une fonction anonyme non vectorielle lui est appliquée.

`map_2()` traite deux arguments. Ici, la fonction est vectorielle, donc `mutate(x^y)` serait plus simple et efficace.

```
tibble(x = 1:5, y = 6:2) %>%
  mutate(puissance = map2_dbl(.x, .y, \(x, y) x^y))
```

```
# A tibble: 5 x 3
      x     y puissance
  <int> <int>     <dbl>
1     1     6         1
2     2     5        32
3     3     4        81
4     4     3        64
5     5     2       25
```

Il n'existe pas de fonction `map3()` pour traiter plus de deux arguments.