The Map Function in R

Há situações onde queremos aplicar uma mesma função várias vezes, e ao invés de repetir código chamando a mesma função várias vezes, a função `map` da biblioteca **purrr** pode, além de economizar linhas de código, vetorizar a operação tornando o código mais rápido e eficiente.

```
library(purrr)

vetor <- c(1,2,3,4,5)

soma_um <- function(num){
  return(num+1)
}

saida <- map(vetor, soma_um)
saida
```

```
## [[1]]
## [1] 2
##
## [[2]]
## [1] 3
##
## [[3]]
## [1] 4
##
## [[4]]
```

```
## [1] 5
##
## [[5]]
## [1] 6
```

Convertendo Lista para Vetor

No entanto um problema do resultado dessa função, é que o resultado fica num formato de lista, diferente do formato de vetor inicial. A função `unlist` converte de volta para o formato de vetor.

```
unlist(saida)
```

```
## [1] 2 3 4 5 6
```

map2

A Função `map` apesar de muito útil tem uma limitação de aceitar um único valor na entrada. Foi pensando nisso que criaram a função `map2` que consegue lidar com 2 valores.

```
vetor2 <- c(10,9,8,7,6)

soma_dois <- function(num1, num2){
  return(num1 + num2)
}

saida2 <- map2(vetor, vetor2, soma_dois) %>% unlist()
saida2
```

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

tibble

Até o momento abordamos o uso do map com vetores, mas não se limita a isso, sendo muito útil ao aplicar em colunas de um tibble.

No exemplo também demonstramos que é possível simplesmente somar a+b, mas a questão está na eficiência de processamento principalmente em larga escala.

```
dataset <- tibble(
  a= 1:5,
  b= 10:6
)

dataset <- dataset %>% mutate(c = map2(a,b,soma_dois) %>%
                                  unlist(),
                              d = a+b)
dataset
```

```
## # A tibble: 5 x 4
##       a     b     c     d
##   <int> <int> <int> <int>
## 1     1    10    11    11
## 2     2     9    11    11
## 3     3     8    11    11
## 4     4     7    11    11
## 5     5     6    11    11
```

Na documentação da biblioteca purrr que você pode encontrar clicando aqui, existe uma infinidade de variações do map cada um atendendo uma situação específica.

Agora para finalizar sobre a função `map`, um exemplo que demonstra o seu real valor. Um tibble é na verdade um conjunto de listas (cada coluna uma lista), e a função map além de vetores aceita listas. Então basicamente conseguimos fazer uma operação com um dataset inteiro apenas com uma chamada da função.

```
map(dataset, sum) %>% unlist()
```

```
##  a  b  c  d
## 15 40 55 55
```

summary, group_by e summarise

A função `summary` é uma alternativa para de forma rápida e ainda vetorizada descrever o conteúdo de um dataset. Por padrão alguns cálculos são feitos, como o valor mínimo, máximo, média e mediana de cada coluna.

Também é possível entender um pouco mais a fundo a distribuição desses valores, se tem valores variados ou muito concentrados em certos valores, isso através do 1st e 3rd Quarter. Em outras palavras o 1st Quarter é o primeiro quarto da base, os primeiros 25% da base de forma ordenada se concentram até o valor 6,75 na coluna nota.

```
aulas <- tibble(
  aluno   = c("maria","joao","raimundo","mariana",
              "cristiano","rafaela","julia","gabriel",
              "maria","joao","raimundo","mariana",
              "cristiano","rafaela","julia","gabriel"),
  materia = c("biologia", "biologia","biologia","biologia",
              "biologia","biologia","biologia","biologia",
              "portugues","portugues","portugues","portugues",
              "portugues","portugues","portugues","portugues"),
  nota    = c(10, 9, 10, 8, 8, 5, 8, 6,
              7, 6, 4, 8, 8, 7, 10, 9)
)

aulas %>% summary()
```

```
##     aluno              materia               nota      
##  Length:16          Length:16          Min.   : 4.000  
##  Class :character   Class :character   1st Qu.: 6.750  
##  Mode  :character   Mode  :character   Median : 8.000  
##                                        Mean   : 7.688  
##                                        3rd Qu.: 9.000  
##                                        Max.   :10.000
```

O map até o momento aceita uma ou duas entradas e de forma vetorizada faz o processamento e devolve uma saída. E o summary pode trazer num único comando várias informações que ajudam a entender o conteúdo desse conjunto de dados.

No entanto pode ser necessário várias saídas com regras e cálculos distintos. É aí que entra a função `summarise` que pode ser combinada com o `group_by` e potencializar a análise. Com essas funções conseguimos agrupar os resultados, aplicar funções da forma que for necessário para a análise. Nesse caso é possível saber a nota média da turma por matéria, ou a menor e maior nota de cada aluno por exemplo.

```
aulas %>%
  group_by(materia) %>%
  summarise(media = mean(nota))
```

```
## # A tibble: 2 x 2
##   materia   media
##   <chr>     <dbl>
## 1 biologia   8   
## 2 portugues  7.38
```

```
aulas %>%
  group_by(aluno) %>%
  summarise(minimo = min(nota),
            maximo = max(nota))
```

```
## # A tibble: 8 x 3
##   aluno     minimo maximo
##   <chr>      <dbl>  <dbl>
## 1 cristiano      8      8
## 2 gabriel        6      9
## 3 joao           6      9
## 4 julia          8     10
## 5 maria          7     10
## 6 mariana        8      8
## 7 rafaela        5      7
## 8 raimundo       4     10
```