# Iteration

Não precisamos copiar e colar um código para executá-lo várias vezes, até porque qualquer alteração teria de ser replicada para todas as cópias dificultando a manutenção. Uma estrutura muito comum para resolver esse problema é a Iteração.

Aqui veremos 2 tipos de Iteração, estrutura `for` e `while`.

## Estrutura `for`

Imagine que o time de marketing solicitou para calcular o lucro de vários produtos com a seguinte fórmula:

Lucro = Preço - (Preço * Impostos%) - (Preço * Custo%)

Sem uma estrutura de Iteração o cálculo ficaria assim.

```
precos <- c(150.00, 145.00, 180.00, 199.99, 200.99)
lucros <- c(
  150.00 - (150.00 * 0.20) - (150.00 * 0.10),
  145.00 - (145.00 * 0.20) - (145.00 * 0.10),
  180.00 - (180.00 * 0.20) - (180.00 * 0.10),
  199.99 - (199.99 * 0.20) - (199.99 * 0.10),
  200.99 - (200.99 * 0.20) - (200.99 * 0.10)
  )
lucros
```

```
## [1] 105.000 101.500 126.000 139.993 140.693
```

Agora com uma estrutura de repetição, apenas escrevemos a fórmula uma vez utilizando variáveis para armazenar os valores.

```
lucros <- c()

for (preco in precos){
  lucro <- preco - (preco * 0.20) - (preco * 0.10)
  lucros <- c(lucros, lucro)
}
lucros
```

```
## [1] 105.000 101.500 126.000 139.993 140.693
```

Podemos ler o código acima da seguinte forma:

Para cada preço na lista de preços calcule a seguinte fórmula e armazene na variável lucro. Em seguida adicione o resultado de lucro no final da lista de lucros.

O que delimita a quantidade de repetições é o tamanho do vetor preços, 10 preços geraria 10 execuções da estrutura.

Outra forma de executar o mesmo código acima é de forma **indexada**.

```
lucros <- integer(length(precos))
indices <-  1:length(precos)

for (i in indices){
  lucro <- precos[i] - (precos[i] * 0.20) - (precos[i] * 0.10)
  lucros[i] <- lucro
}
lucros
```

```
## [1] 105.000 101.500 126.000 139.993 140.693
```

No trecho acima a variável i assume um valor a cada rodada que vai de 1 a 5, e assim acessa a posição 1 do vetor `precos`, em seguida a posição 2 e assim por diante.

## Estrutura `while`

Na estrutura for, podemos de forma metafórica comparar com uma lista de atividades, sabemos previamente quantas rodadas serão necessárias para finalizar, é como uma lista de tarefas. Pensando no exemplo anterior, tínhamos apenas 5 preços, portanto só era necessário calcular o lucro 5 vezes.

Mas há casos onde é necessário atingir uma "meta" e não sabemos quantas execuções serão necessárias para chegar lá, sendo assim aqui temos o cenário perfeito para o uso do while.

```
nums <- 1:10
condicao_eh_verdadeira <- FALSE
i <- 1

while (!condicao_eh_verdadeira) {

  # Busca o núm na posição i do vetor
  num_atual <- nums[i]
  # Confere se atingiu a meta
  condicao_eh_verdadeira <- num_atual > 7
  # Acrescenta em um para olhar a próxima posição
  i = i + 1
}
```

O código acima pode ser lido da seguinte forma:

Enquanto a condição não é verdadeira, continue executando o trecho.

É muito importante ter uma linha de código que controle a variável i garantindo que a execução não fique presa em um loop, sem a adição na variável i, o trecho ficaria preso olhando sempre a primeira posição e não atendendo a condição para sair do loop, executando eternamente.

Agora imagine que o mesmo time de marketing espera ser notificado quando atingir 400 reais de lucro.

```
atingiu_lucro <- FALSE
i <- 1

lucro_garantido <- 0

while (!atingiu_lucro) {

  lucro_garantido <- lucro_garantido +lucros[i]
  atingiu_lucro <- lucro_garantido >= 400
  i = i + 1
  print(lucro_garantido)
}
```

```
## [1] 105
## [1] 206.5
## [1] 332.5
## [1] 472.493
```

```
lucro_garantido
```

```
## [1] 472.493
```

O trecho acima é executado até que a meta de lucro de 400 reais seja atingido, o trecho `lucro_garantido >= 400` quando o lucro for maior ou igual a 400 vai resultar em verdadeiro, e assim atribuir o valor `TRUE` à variável fazendo com que saia do loop.