# Introdução a Linguagem R

Perfil

*Delermando Branquinho Filho*

## Por que o meu código é tão lento?

- Profiling é uma forma sistemática de examinar quanto tempo é gasto em diferentes partes de um programa
- Útil ao tentar otimizar seu código
- Muitas vezes o código funciona bem uma vez, mas e se você tiver que colocá-lo em um loop para 1.000 iterações? Ainda é rápido o suficiente?
- Profiling é melhor do que adivinhar

## Sobre otimizar seu código

- Obter maior impacto na aceleração do código depende de saber onde O código passa a maior parte do seu tempo
- Isso não pode ser feito sem análise de desempenho ou criação de perfil

Devemos esquecer as pequenas eficiências, digamos cerca de 97% Tempo: a otimização prematura é a raiz de todo o mal </ Blockquote>

## Princípios Gerais de Otimização

- Design primeiro, depois otimizar
- Lembre-se: otimização prematura é a raiz de todo o mal
- Medir (coletar dados), não adivinhar.
- Se você vai ser cientista, você precisa aplicar o mesmos princípios aqui!

---

## Usando `system.time ()`

- Toma uma expressão R arbitrária como entrada (pode estar entre chaves) e retorna o tempo necessário para avaliar a expressão
- Calcula o tempo (em segundos) necessário para executar uma expressão
- Se houver um erro, fornece o tempo até que o erro ocorra
- Retorna um objeto da classe `proc_time`
- ** tempo do usuário **: tempo carregado na (s) CPU (s) para esta expressão
- ** tempo decorrido **: tempo do “relógio de parede”

---

## Usando `system.time ()`

- Normalmente, o tempo do usuário eo tempo decorrido são relativamente próximos, para tarefas de computação direta
- O tempo decorrido pode ser * maior que * tempo do usuário se a CPU gasta muito tempo esperando ao redor
- O tempo decorrido pode ser * menor que * o tempo do usuário se sua máquina tem vários núcleos / processadores (e é capaz de usá-los)
- Bibliotecas BLAS multi-threaded (vecLib / Accelerate, ATLAS, ACML, MKL)
- Processamento paralelo via pacote **parallel**

---

## Usando `system.time ()`

```
system.time(readLines("http://www.thescientist.com.br"))
```

```
## Warning in readLines("http://www.thescientist.com.br"): linha final
## incompleta encontrada em 'http://www.thescientist.com.br'
```

```
##    user  system elapsed
##    0.05    0.07    0.50
```

```
myFunction <- function(n) {
        i <- 1:n
        1 / outer(i - 1, i, '+')
        }
x <- myFunction(1000)
system.time(svd(x))
```

```
##    user  system elapsed
##    2.75    0.03    2.82
```

---

## Timing e Expressões mais longas

```
system.time({
        n <- 1000
        r <- numeric(n)
        for(i in 1:n) {
                x <- rnorm(n)
                r[i] <- mean(x)
        }
})
```

```
##    user  system elapsed
##    0.07    0.02    0.10
```

## Além de `system.time ()`

- Usando `system.time ()` permite testar certas funções ou códigos em blocos para ver se eles estão tomando quantidades excessivas de tempo
- Assume que você já sabe onde está o problema e pode ligar `System.time ()` sobre ele
- E se você não sabe por onde começar?

---

## O R Profiler

- A função `Rprof ()` inicia o perfilador em R
- R deve ser compilado com suporte de perfilador (mas isso geralmente é o caso)
- A função `summaryRprof ()` resume a saída de `Rprof ()` (Caso contrário, não é legível)
- NÃO use `system.time ()` e `Rprof ()` juntos ou você ficará triste

---

## O R Profiler

- Rprof () acompanha a pilha de chamadas de função regularmente amostradas em intervalos e tabula quanto tempo é gasto em cada função
- O intervalo de amostragem padrão é de 0,02 segundos
- NOTA: Se o seu código é executado muito rapidamente, o profiler não é útil, mas então você provavelmente não precisa dele nesse caso

## Using `summaryRprof()`

- A função `summaryRprof ()` tabula a saída R calcula quanto tempo é gasto em que função
- Existem dois métodos para normalizar os dados
- "By.total" divide o tempo gasto em cada função pelo tempo de execução total
- "By.self" faz o mesmo, mas primeiro subtrai o tempo gasto em funções acima na pilha de chamada

---

## By Total

```
$by.total
                        total.time total.pct self.time self.pct
"lm"                          7.41    100.00      0.30     4.05
"lm.fit"                      3.50     47.23      2.99    40.35
"model.frame.default"         2.24     30.23      0.12     1.62
"eval"                        2.24     30.23      0.00     0.00
"model.frame"                 2.24     30.23      0.00     0.00
"na.omit"                     1.54     20.78      0.24     3.24
"na.omit.data.frame"          1.30     17.54      0.49     6.61
```

```
"lapply"                    1.04     14.04      0.00     0.00
"[.data.frame"              1.03     13.90      0.79    10.66
"["                         1.03     13.90      0.00     0.00
"as.list.data.frame"        0.82     11.07      0.82    11.07
"as.list"                   0.82     11.07      0.00     0.00
```

---

## By Self

```
$by.self
                        self.time self.pct total.time total.pct
"lm.fit"                     2.99    40.35       3.50     47.23
"as.list.data.frame"         0.82    11.07       0.82     11.07
"[.data.frame"               0.79    10.66       1.03     13.90
"structure"                  0.73     9.85       0.73      9.85
"na.omit.data.frame"         0.49     6.61       1.30     17.54
"list"                       0.46     6.21       0.46      6.21
"lm"                         0.30     4.05       7.41    100.00
"model.matrix.default"       0.27     3.64       0.79     10.66
"na.omit"                    0.24     3.24       1.54     20.78
"as.character"               0.18     2.43       0.18      2.43
"model.frame.default"        0.12     1.62       2.24     30.23
"anyDuplicated.default"      0.02     0.27       0.02      0.27
```

---

## `summaryRprof()` Output

```
$sample.interval
[1] 0.02

$sampling.time
[1] 7.41
```

---

## Resumo

- `Rprof ()` executa o perfilador para o desempenho da análise do código R
- `SummaryRprof ()` resume a saída de `Rprof ()` e fornece o percentual do tempo gasto em cada função
- Bom para quebrar seu código em funções para que o perfilador possa dar informações úteis sobre onde o tempo está sendo gasto
- Código C ou Fortran não é perfilado