

Análise dos Dados com R

Styling e Linting

James R. Hunter, PhD

Retrovirologia, EPM, UNIFESP

2024-11-19

Estilo de Código

Estilo Conta!

- Escrevemos código não só para ser executado
- Pessoas precisam ler
- Você precisa ler
 - Até meses, anos depois
- Precisa ter sentido

Tidyverse Style Guide

- Cópia no Github
- Existem várias guias para estilo correto
- São todas relacionadas
 - Google, Tidyverse, outros
- Gosto de Tidyverse; vamos usar este

Pacote **styler**

- Instalado com os pacotes iniciais
- Baseado na guia do estilo de Tidyverse
- Examina um arquivo
- Faz as mudanças necessárias para colocar ele em um estilo bom.

Exemplo de `styler::style_file()`

- Código fora de padrão

```
1 A=c(6,10,12,14,16,20,22,24,26,30)
2 b=c("gato", "cachorro", "peixe", "rato", "galinha")
3 x=list(a, b)
4
5 sumA=sum(A)
6 div_a=A[4]/A[1]
```

Linhas # 1 & 2

```
1 A=c(6,10,12,14,16,20,22,24,26,30)
2 b=c("gato", "cachorro", "peixe", "rato", "galinha")
```

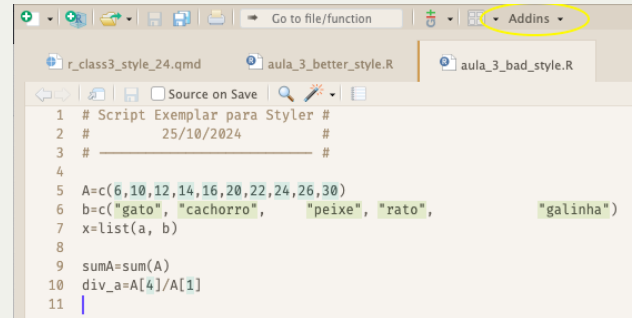
- ■ “=” invés de “<-”
 - Irregular - espaço branco

Linhas # 4 & 5

```
1 sumA=sum(A)
2 div_a=A[4]/A[1]
```

- “=” invés de “<-”
- Espaço branco na divisão em volta da “/”

Como Usar **styler**



1. Clicar nos **Addins**
2. Rolar para baixo para área de **STYLER**
3. Clicar no “**Style Active File**”
4. Styler vai fazer as correções dos erros que ele conhece
 - *Automático*
5. Salvar o arquivo com um novo nome

Nosso Arquivo com Correções

```
1 A <- c(6, 10, 12, 14, 16, 20, 22, 24, 26, 30)
2 b <- c("gato", "cachorro", "peixe", "rato", "galinha")
3 x <- list(a, b)
4
5 sumA <- sum(A)
6 div_a <- A[4] / A[1]
```

Linting

O Que É

- Indica o que são os erros de formatação do código
- Comentários muito mais detalhados que **styler**
- Mas, não faz as mudanças
- Produz um relatório no window de “Markers” com os comentários
- Executar da mesma maneira que “Style Active File” com o “Addin” **Lint**

Resultado - Lint

```
lintr ▾
~/Documents/MAD/Bioinformatica Aplicada 2024/aulas_2024/class_notes/aula_3_bad_style.R
$ Line 2 [commented_code_linter] Commented code should be removed.
$ Line 5 [object_name_linter] Variable and function name style should match snake_case or symbols.
$ Line 5 [assignment_linter] Use ←, not =, for assignment.
$ Line 5 [infix_spaces_linter] Put spaces around all infix operators.
$ Line 5 [commas_linter] Commas should always have a space after.
$ Line 5 [commas_linter] Commas should always have a space after.
$ Line 5 [commas_linter] Commas should always have a space after.
$ Line 5 [commas_linter] Commas should always have a space after.
$ Line 5 [commas_linter] Commas should always have a space after.
$ Line 5 [commas_linter] Commas should always have a space after.
$ Line 5 [commas_linter] Commas should always have a space after.
$ Line 5 [commas_linter] Commas should always have a space after.
$ Line 5 [commas_linter] Commas should always have a space after.
$ Line 6 [assignment_linter] Use ←, not =, for assignment.
$ Line 6 [infix_spaces_linter] Put spaces around all infix operators.
$ Line 7 [assignment_linter] Use ←, not =, for assignment.
$ Line 7 [infix_spaces_linter] Put spaces around all infix operators.
$ Line 9 [object_name_linter] Variable and function name style should match snake_case or symbols.
$ Line 9 [assignment_linter] Use ←, not =, for assignment.
$ Line 9 [infix_spaces_linter] Put spaces around all infix operators.
$ Line 10 [assignment_linter] Use ←, not =, for assignment.
$ Line 10 [infix_spaces_linter] Put spaces around all infix operators.
$ Line 10 [infix_spaces_linter] Put spaces around all infix operators.
```