



1



2

Funções Apply

- Conjunto de funções usadas para aplicar operações em uma coleção de dados
- Objetivo primário é evitar o uso de laços
- Aplicados em vetores, matrizes, listas ou data frames
- Funções
 - `lapply()`
 - `sapply()`
 - `apply()`
 - `tapply()`
 - ...

Prof. Dr. Razer A N R Montañó

SEPT/UFR

3

3

Funções Apply: `sapply`

- A função **`sapply()`** efetua um laço sobre os elementos passados e aplica uma função a cada elemento
- Retorna um resultado simplificado (vetor ou matriz)

```
sapply(<vetor ou lista>, <função>, <argumentos...>)
```

- Exemplo

```
> x <- 10:20
```

```
> sapply(x, sqrt)
```

```
[1] 3.162278 3.316625 3.464102 3.605551 3.741657 3.872983
4.000000 4.123106
```

```
[9] 4.242641 4.358899 4.472136
```

Prof. Dr. Razer A N R Montañó

SEPT/UFR

4

4

Funções Apply: **sapply**

- Pode-se usar uma função customizada (UDF) para o **sapply()**

```
sapply(<vetor ou lista>, <função>, <argumentos...>)
```

- Exemplo

```
quadrado <- function(x) {
  return (x^2)
}

dados <- 10:20

sapply(dados, quadrado)

[1] 100 121 144 169 196 225 256 289 324 361 400
```

Funções Apply: **sapply**.

- Se a função a ser aplicada tem mais de um parâmetro, pode-se passá-lo na chamada de **sapply()**

```
potencia <- function(x, expoente) {
  return (x^expoente)
}

dados <- 10:20

sapply(dados, potencia, expoente=4)

[1] 10000 14641 20736 28561 38416 50625 65536
83521 104976 130321 160000
```

Funções Apply: lapply.

- A função **lapply()** funciona do mesmo jeito que **sapply()**, mas retorna uma lista

- Exemplo

```
> x <- 10:20
> lapply(x, sqrt)
[[1]]
[1] 3.162278

[[2]]
[1] 3.316625

[[3]]
[1] 3.464102

[[4]]
[1] 3.605551
...
```

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

7

7

Funções Apply: apply.

- A função **apply()** efetua a varredura em uma matriz/data frame por linha ou por coluna

- Margem: 1 – por linha, 2 – por coluna

```
apply(<matriz>, <margem>, <função>, <... Argumentos>)
```

- Exemplo

```
> matriz <- matrix(1:12, nrow=3, ncol=4)
> matriz
      [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> apply(matriz, 1, sum)
[1] 22 26 30
> apply(matriz, 2, sum)
[1]  6 15 24 33
```

Soma cada uma das linhas

Soma cada uma das colunas

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

8

8

Funções Apply: tapply

- A função **tapply()** efetua a operação sobre os dados, mas de forma agrupada por um vetor de fatores (categorias)

```
tapply(<dados>, <INDEX>, <FUN>, <Argumentos>)
```

- Exemplo

```
> tapply(iris$Sepal.Width, iris$Species, mean)
setosa versicolor virginica
3.428    2.770    2.974
```

- Descrição
 - Está calculando a média do atributo **Sepal.Width** da base iris
 - Está calculando agrupado pelo atributo **Species**
 - O resultado é agrupado por **Species**

Funções Apply: tapply

- Outro exemplo: criação da base de dados

```
> n_alunos <- 200
> n_cursos <- 4
> v_nome <- paste("Aluno", 1:n_alunos)
> v_nota <- sample(0:100, n_alunos, replace=T)
> v_cursos <- paste("Curso", 1:n_cursos)
> v_curso <- v_cursos[sample(1:n_cursos, n_alunos, replace=T)]

> df <- data.frame(nome=v_nome, nota=v_nota, curso=v_curso)
> head(df)
  nome nota  curso
1 Aluno 1  95 Curso 3
2 Aluno 2  59 Curso 4
3 Aluno 3  72 Curso 1
4 Aluno 4  43 Curso 2
5 Aluno 5  83 Curso 1
6 Aluno 6  83 Curso 4
```

Funções Apply: **tapply**.

- Outro exemplo: aplicação do **tapply()**

```
> head(df)
```

```
      nome nota  curso
1 Aluno 1   95 Curso 3
2 Aluno 2   59 Curso 4
3 Aluno 3   72 Curso 1
4 Aluno 4   43 Curso 2
5 Aluno 5   83 Curso 1
6 Aluno 6   83 Curso 4
```

```
> tapply(df$nota, df$curso, mean)
```

```
Curso 1 Curso 2 Curso 3 Curso 4
56.00000 46.20370 64.05263 42.08475
```

Apply Paralelo.

- Aplicação da função **apply** em paralelo: **mclapply()**
 - <https://www.rdocumentation.org/packages/parallel/versions/3.4.1/topics/mclapply>
 - <https://dept.stat.lsa.umich.edu/~jerrick/courses/stat701/notes/parallel.html>



Exercícios.

1. Execute os exercícios apresentados nos slides
2. Crie uma matriz com 10 colunas contendo 100.000 números, sendo os números de 1:100000.
 - Execute um comando `apply` que calcula a soma de cada linha desta matriz.
 - Execute um comando `apply` que calcula a média de cada coluna desta matriz.
3. Crie o seguinte data frame

```
> idade <- c(56, 34, 67, 33, 25, 28)
> peso  <- c(78, 67, 56, 44, 56, 89)
> altura <- c(165, 171, 167, 167, 166, 181)
> dados <- data.frame(idade, peso, altura)
```

 - Dê as seguintes respostas
 - A média de todas as colunas (usando `apply`)
 - O valor máximo de todas as colunas (usando `apply`)
 - A raiz quadrada de todos os valores do data frame, como uma matriz
 - A raiz quadrada de todos os valores do data frame, como uma lista
 - Todos os valores do data frame multiplicados por 20, como uma matriz (usando uma UDF)