Manipulando objetos no R

Andreas Kneip

24 de abril de 2018

O R

Para se acostumar com a linguagem, siga as instruções deste tutorial que está na página oficial do

Objetivos

- Aprender como gerenciar o ambiente de trabalho no R
- Conhecer os tipos de entrada e saída do R
- Conhecer os objetos do R
- Aprender como manipular os objetos no R

Diretório de trabalho

As funções criadas por você e que serão utilizadas em uma seção do R deve estar no diretório corrente.

O diretório corrente pode ser determinada com a função

```
getwd()
```

O diretório corrente pode ser alterado usando a função setwd(). Exemplo:

```
setwd('~/Documentos/uft/mestrado_computacao/2018_1/Dados/')
```

Pacotes no R

- A funcionalidade do R pode ser estendida utilizando pacotes, que são funções e procedimentos desenvolvido por alguns usuários e que são disponibilizados para outros usuários.
- Para verificar quais pacotes estão instalados na sua máquina, use

```
library()
```

Instalando pacotes no R

- Descubra qual pacote possui a funcionalidade que você está procurando em https://cran.r-project.org/. Encontrado o pacote desejado, por exemplo, o pacote ggplot2, instale com

```
install.packages("ggplot2")
```

- Você será solicitado a escolher um site espelho para fazer o download
- Carregue o pacote recém instalado com

```
library(ggplot2)
```

ou com

```
require(ggplot2)
```

Pacotes ativos

Para saber quais pacotes estão ativos em uma sessão no R use

```
search()
```

```
[1] ".GlobalEnv"        "package:stats"     "package:graphics"
[4] "package:grDevices" "package:utils"     "package:datasets"
[7] "package:methods"   "Autoloads"         "package:base"
```

Observações

- Caracteres e strings de caracteres devem vir dentro de “aaaa” ou ‘aaaa’
- Se uma palavra não estiver dentre de “aaaa” ou ‘aaaa’ o R vai entender que você está se referindo um objeto. Se este objeto existir, ele vai ser carregado. Se não existir, o R dará uma mensagem de erro
- O R faz distinção entre maiúsculas e minúsculas
- O R é uma linguagem orientada a objetos, assim, possui diferentes métodos para diferentes tipos de objetos

Salvando e carregando objetos no R

Salvar os objetos:

- save(objeto, file = “meusObjetos.RData”)

Salvando todos os objetos da sessão:

- save(list = ls(all=TRUE), file= “tudo.Rdata”)

Carregar objetos de um arquivo:

- load(“meusObjetos.RData”)

Entrada e saída no R

Entrada:

- É possível carregar uma série de comandos de um arquivo: source(“comandos.r”)

Saída:

- É possível redirecionar a saída para um arquivo: sink(“saida.lis”)

Para voltar com a saída em vídeo: sink()

Tipos de saída

- PDF: pdf(“filename.pdf”)
- Windows Metafile: win.metafile(“filename.wmf”)
- PNG: png(“filename.png”)
- JPEG: jpeg(“filename.jpg”)
- BMP: bmp(“filename.bmp”)
- Postscript: postscript(“filename.ps”)

Mais sobre entradas e saídas

Pode-se carregar um arquivo texto em um objeto:

- medidas <- scan(file = “analise.txt”);

ou ainda:

- medidas <- scan(file=“analise.csv”, sep=“,”)

O scan também serve para ler do teclado (finalize com 2 toques na tecla enter):

- notas <- scan()

Importando dados de uma planilha

Crie a planilha

- Salve como tipo CSV: PlanR.csv

Table 1: PlanR.csv

Aluno	Prova1	Prova2
Ari	9	7
Eliana	8	8
Zulmira	5	7

Importando a planilha

```
plan <- read.csv("~/PlanR.csv", header=TRUE, sep=",", row.names=1)
plan
```

```
        Prova1 Prova2
Ari          9      7
Eliana       8      8
Zulmira      5      7
```

Obtendo algumas informações

Média por aluno:

```
apply(plan,1,mean)
```

```
    Ari  Eliana Zulmira
      8       8       6
```

Sumário das provas:

```
apply(plan,2,summary)
```

```
          Prova1   Prova2
Min.    5.000000 7.000000
1st Qu. 6.500000 7.000000
Median  8.000000 7.000000
Mean    7.333333 7.333333
3rd Qu. 8.500000 7.500000
Max.    9.000000 8.000000
```

Exemplos de E/S

- source(“arquivo1”)
- sink(“arqSaida”, append=TRUE, split=TRUE)
- pdf(“saidaPDF.pdf”)
- source(“arquivo2”)
- sink()
- dev.off()
- source(“arquivo3”)

Ambiente de trabalho

- Diretório corrente: getwd()

- Mudar o diretório: setwd("/diretorio")
- Objetos na memória: ls()
- Seus últimos 20 comandos: history(20)
- Gravar ou carregar histórico:
  - savehistory("arquivo")
  - loadhistory("arquivo")
- Ver as opções: options()

Modos e atributos de um objeto

- O objeto mais simples do R é um vetor
- Mesmo uma variável (x = 2 por exemplo), é considerada como um vetor de tamanho 1
- Concatenar / combinar vetores:

$$a = c(1, 3, 5); b = c(2, 4, 6)$$

$$d = c(a, b)$$

  Pode-se acessar elementos do vetor com colchetes:

$$d[3] = 5$$

Aritmética de vetores

Realize as seguintes operações:

$$v = c(1, 2, 3, 4, 5)$$

$$2 * v$$

$$w = c(1)$$

$$v * w; v + w$$

$$w = c(1, 2)$$

$$v * w; v + w$$

$$c(w, v, w)$$

Funções embutidas

- sqrt: sqrt(2) = 1.4142136
- log: log(1024, 2) = 10
- exp: exp(2) = 7.3890561
- pi = 3.1415927
- sin, cos, tan, asin, acos, atan: sin(pi/4) = 0.7071068
- abs(-17) = 17
- Tente: 13/0; sqrt(-17); sqrt(-17 + 0i)

Outras operações

- exponenciação: 2^10 = 1024
- resto da divisão: 7 %% 3 = 1
- inteiro da divisão: 7 %/% 3 = 2

Funções para vetores

Exemplo:

```
vet <- c(1, 4, 7, 2, 5, 8)
```

- Intervalo: range(vet) = 1, 8
- Comprimento: length(vet) = 6
- Somatório: sum(vet) = 27
- Soma acumulada: cumsum(vet) = 1, 5, 12, 14, 19, 27
- Produtório: prod(vet) = 2240
- Produtório acumulados: cumprod(vet) = 1, 4, 28, 56, 280, 2240

Funções sumário

- Média: mean(vet) = 4.5
- Variância: var(vet) = 7.5
- Máximo e mínimo: max(vet) = 8, min(vet) = 1
- Sumário: summary(vet)

```
summary(vet)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
    1.0     2.5     4.5     4.5     6.5     8.0 
```

Ordenação de vetores

- Ordem inversa:

```
rev(vet)
```

```
[1] 8 5 2 7 4 1
```

Ordenação:

```
sort(vet)
```

```
[1] 1 2 4 5 7 8
```

- Tente:

```
sort(vet, decreasing = TRUE)
```

Outras funções para vetores

- Obter um vetor de permutações que ordenam o vetor dado:

```
order(vet)
```

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

- Posto dos valores do vetor:

```
rank(vet)
```

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

- Compare rank com order

Intervalo e sequências

- Intervalo unitário de 1 a 10:

```
1:10
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

- Gere um intervalo com 10 números, de 2 até 20

```
seq(2,20,2)
```

```
 [1]  2  4  6  8 10 12 14 16 18 20
```

Experimente

- Faça: n = 10; 1:n-1; 1:(n-1)
  - seq(from = 1, to = 10) # from e to são opcionais
  - seq(1, 10, by = 2)
  - seq(1,10, length = 3)
  - seq(length = 51, from = -5, by = .2)

Repetições

```
x = 1:3
rep(x, times = 5)
```

```
 [1] 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3
```

```
rep(x, each = 5)
```

```
 [1] 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3
```

Mais repetições

```
y = rep( c(4, 2, 8, 10, 6), c(1, 2, 2, 1, 3) )
y
```

```
[1]  4  2  2  8  8 10  6  6  6
```

```
unique(y)
```

```
[1]  4  2  8 10  6
```

```
duplicated(y); y[!duplicated(y)]
```

```
[1] FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE
[1]  4  2  8 10  6
```

Exercício

Construa um vetor com os números de 14 a 7 seguido de 7 números 4 seguido de 8 até 16 passo 2.

Valores lógicos

TRUE, FALSE ou NA

```
hip = 3 < 2; hip
```

```
[1] FALSE
```

```
x = 1:3; x <= 2
```

```
[1]  TRUE  TRUE FALSE
```

== igual, != diferente, & e, | ou, ! negação

Cuidado com T e F!

Mais sobre valores lógicos

```
z <- c(1:3, NA); z
```

```
[1]  1  2  3 NA
```

```
ind <- is.na(z); ind
```

```
[1] FALSE FALSE FALSE  TRUE
```

Mais ainda sobre valores lógicos

Não confunda com NA com NaN:

```
0/0; Inf-Inf; is.nan(NA); is.na(NaN)
```

```
[1] NaN
```

```
[1] NaN
```

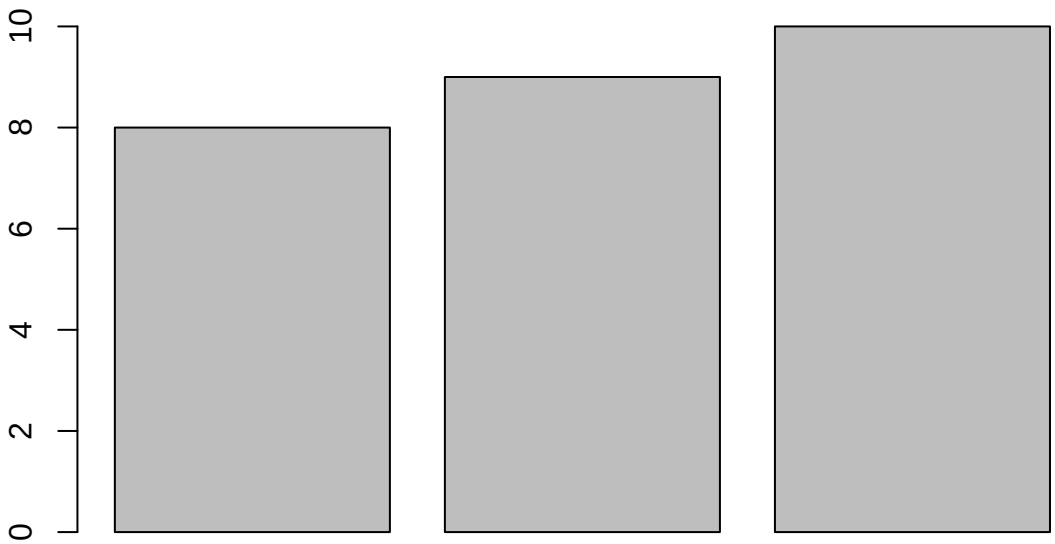
```
[1] FALSE
```

```
[1] TRUE
```

Vetores de caracteres

Também usados em gráficos

```
barplot(8:10, main = "Notas da Prova")
```



Caracteres

- Caractere é definido por " ou por '
- Caracteres especiais são precedidos de \
- Para consultá-los:

```
?Quotes
```

- Concatenar vetores transformando para strings:

```
paste(0:1, "berto", sep="")
```

```
[1] "0berto" "1berto"
```

```
paste(c("Al","Ro","Gil","Dago"), "berto", sep="")
```

```
[1] "Alberto"   "Roberto"   "Gilberto"  "Dagoberto"
```

Índices de vetores

Pode-se acessar um elemento do vetor:

```
v = 1:10; v
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

```
v[5] = NA; v
```

```
 [1]  1  2  3  4 NA  6  7  8  9 10
```

```
y <- v[!is.na(v)]; y
```

```
[1]  1  2  3  4  6  7  8  9 10
```

Mais sobre índices de vetores

```
(v + 1)[(!is.na(v)) & v > 5] -> z; z
```

```
[1]  7  8  9 10 11
```

```
v[4:6]
```

```
[1]  4 NA  6
```

```
c("x","y")[rep(c(1, 2, 2, 1), times = 4)]
```

```
 [1] "x" "y" "y" "x" "x" "y" "y" "x" "x" "y" "y" "x" "x" "y" "y" "x"
```

```
y <- v[-(4:6)]; y
```

```
[1]  1  2  3  7  8  9 10
```

Mais ainda sobre índices de vetores

Acessando diversos intervalos de índices:

```
x <- 1:10; x[c(2:3, 5, 7:8)] <- NA; x
```

```
 [1]  1 NA NA  4 NA  6 NA NA  9 10
```

Trocando NA por 0:

```
x[is.na(x)] <-0; x
```

```
 [1]  1  0  0  4  0  6  0  0  9 10
```

O que este comando faz?

```
y = 5:-5; y
```

```
 [1]  5  4  3  2  1  0 -1 -2 -3 -4 -5
```

```
y[y < 0] <- -y[y < 0]
```

Resposta

```
y = 5:-5; y
```

```
 [1]  5  4  3  2  1  0 -1 -2 -3 -4 -5
```

```
y[y < 0] <- -y[y < 0]; y
```

```
 [1] 5 4 3 2 1 0 1 2 3 4 5
```

Atribuindo nomes

Podemos nomear os índices para facilitar a identificação de ítens num vetor

Entre com os nomes: Renata, Lili e Ari. Lembre-se de terminar com dois "enter".

```
aluno <- scan(what = character(3))
```

```
aluno <- c("Renata", "Lili", "Ari")
```

```
nota <- 8:10
names(nota) <- aluno
w <- nota[c("Renata", "Lili")]; w
```

```
Renata   Lili
     8      9
```

Outros tipos de objetos

- matrizes: vetores com mais de um índice
- fatores: para manipulação de dados categorizados
- listas: vetor com elementos não necessariamente do mesmo tipo
- data frames: matrizes com diferentes tipos de colunas
- funções: pequenos programas

Atributos intrínsecos

- vetores possuem só um modo (numérico, complexo, lógico, caractere): mode
- listas tem seu próprio modo, já que um objeto da lista pode ser de qualquer modo, até uma outra lista
- atributos não intrínsecos: attributes #tente no vetor nota criado anteriormente!

```
attributes(nota)
```

```
$names
[1] "Renata" "Lili"   "Ari"
```

Coerção

- Coerção: troca de modo de um objeto
- Pode-se trocar os modos de um vetor:

```
num <- 1:10; num
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

```
as.character(num) -> digito; digito
```

```
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10"
```

```
renum <- as.integer(digito); renum
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

Outras formas de coerção

- Também funciona: as.double, as.numeric
- Exercício: converta para lógico

```
nbool <- c("TRUE", "FALSE"); nbool
```

```
[1] "TRUE"  "FALSE"
```

Solução

```
nbool <- as.logical(nbool); nbool
```

```
[1]  TRUE FALSE
```

Alterando o tamanho

Mesmo um vetor vazio tem um modo:

```
e <- numeric(); length(e)
```

```
[1] 0
```

E seu tamanho pode ser alterado:

```
e[3] = 2; length(e)
```

```
[1] 3
```

Mais

```
a = 20:30; length(a)
```

```
[1] 11
```

```
a = a[2 * 1:5]; length(a)
```

```
[1] 5
```

Para matrizes

```
z = 1:25; z
```

```
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
[24] 24 25
```

Resultado

```
attr(z, "dim") <- c(5,5)
z
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    6   11   16   21
[2,]    2    7   12   17   22
[3,]    3    8   13   18   23
[4,]    4    9   14   19   24
[5,]    5   10   15   20   25
```

Classes

Além de modos, objetos tem classes:

```
x = 1:10; class(x)
```

```
[1] "integer"
```

```
y = 3+1i; class(y)
```

```
[1] "complex"
```

Mais sobre classes

Dependendo da classe, alguns métodos se comportam de forma diferente.

```
summary(x)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   1.00    3.25    5.50    5.50    7.75   10.00 
```

```
summary(y)
```

```
 Length   Class    Mode 
      1 complex complex 
```

Para matrizes

```
class(z) # do exemplo anterior
```

```
[1] "matrix"
```

```
summary(z)
```

```
       V1          V2          V3          V4          V5      
 Min.   :1   Min.   : 6   Min.   :11   Min.   :16   Min.   :21  
 1st Qu.:2   1st Qu.: 7   1st Qu.:12   1st Qu.:17   1st Qu.:22  
 Median :3   Median : 8   Median :13   Median :18   Median :23  
 Mean   :3   Mean   : 8   Mean   :13   Mean   :18   Mean   :23  
 3rd Qu.:4   3rd Qu.: 9   3rd Qu.:14   3rd Qu.:19   3rd Qu.:24  
 Max.   :5   Max.   :10   Max.   :15   Max.   :20   Max.   :25  
```

Fatores

É um vetor de objetos usado para especificar uma classificação discreta (agrupamento)

```
sex <- c("m","m","f","f","f","m","m","f","m","f","f")
sexof <- factor(sex); sexof
```

```
 [1] m m f f f m m f m f f
Levels: f m
```

```
peso <- c(63, 80, 49, 59, 55, 72, 90, 73, 68, 53, 61)
pesomed <- tapply( peso, sexof, mean ); pesomed
```

```
       f        m
58.33333 74.60000
```

Se os fatores tiverem ordem, use ordered ao invés de factor

Matrizes

Matrizes são vetores com mais de um índice

```
mat = 1:150; mat
```

```
  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
 [18]  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34
 [35]  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51
 [52]  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68
 [69]  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85
 [86]  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102
[103] 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119
[120] 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136
[137] 137 138 139 140 141 142 143 144 145 146 147 148 149 150
```

```
dim(mat) = c(15,10)
```

Resultado

```
mat
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1   16   31   46   61   76   91  106  121   136
 [2,]    2   17   32   47   62   77   92  107  122   137
 [3,]    3   18   33   48   63   78   93  108  123   138
 [4,]    4   19   34   49   64   79   94  109  124   139
 [5,]    5   20   35   50   65   80   95  110  125   140
 [6,]    6   21   36   51   66   81   96  111  126   141
 [7,]    7   22   37   52   67   82   97  112  127   142
 [8,]    8   23   38   53   68   83   98  113  128   143
 [9,]    9   24   39   54   69   84   99  114  129   144
[10,]   10   25   40   55   70   85  100  115  130   145
[11,]   11   26   41   56   71   86  101  116  131   146
[12,]   12   27   42   57   72   87  102  117  132   147
[13,]   13   28   43   58   73   88  103  118  133   148
[14,]   14   29   44   59   74   89  104  119  134   149
[15,]   15   30   45   60   75   90  105  120  135   150
```

Matrizes multidimensionais

É possível também criar matrizes com 3 ou mais dimensões:

```
dim(mat) = c(5, 10, 3)
```

Para selecionar elementos:

```
mat[1, 2, 3]
```

```
[1] 106
```

Submatrizes

É possível selecionar uma "subseção", que também é uma matriz:

```
nm <- mat[ , , 2]; nm
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]   51   56   61   66   71   76   81   86   91    96
[2,]   52   57   62   67   72   77   82   87   92    97
[3,]   53   58   63   68   73   78   83   88   93    98
[4,]   54   59   64   69   74   79   84   89   94    99
[5,]   55   60   65   70   75   80   85   90   95   100
```

```
dim(nm)
```

```
[1]  5 10
```

Matrizes

```
mm <- matrix(1:15, nrow = 3, ncol = 5); mm
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10   13
[2,]    2    5    8   11   14
[3,]    3    6    9   12   15
```

```
mm <- matrix(1:15, nrow = 3, ncol = 5, byrow = TRUE);mm
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
[3,]   11   12   13   14   15
```

Mais matrizes

```
mm <- matrix(1:15, nrow = 3, byrow = TRUE); mm
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
[3,]   11   12   13   14   15
```

```
numb = 1:15
mm <- matrix(numb, ncol = 5, byrow = FALSE); mm
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10   13
[2,]    2    5    8   11   14
[3,]    3    6    9   12   15
```

Ainda mais matrizes

Pode-se construir matrizes combinando vetores por coluna ou linha:

```
mat.n1 = 1:3; mat.n2 = 4:6; mat.n3 = 7:9
mat.a = cbind(mat.n1, mat.n2, mat.n3); mat.a
```

```
     mat.n1 mat.n2 mat.n3
[1,]      1      4      7
[2,]      2      5      8
[3,]      3      6      9
```

```
mat.b = rbind(mat.n1, mat.n2, mat.n3); mat.b
```

```
       [,1] [,2] [,3]
mat.n1    1    2    3
mat.n2    4    5    6
mat.n3    7    8    9
```

E ainda mais

E também lendo do teclado:

```
mat3 = matrix(scan(), ncol = 3, byrow = TRUE)
```

Operações com matrizes

```
mar <- matrix(1:16, 4, 4); mar
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
```

Diagonal:

```
diag(mar)
```

```
[1]  1  6 11 16
```

Mais

```
diag(mar) <- 25; mar
```

```
     [,1] [,2] [,3] [,4]
[1,]   25    5    9   13
[2,]    2   25   10   14
[3,]    3    7   25   15
[4,]    4    8   12   25
```

Transposta

```
t(mar)
```

```
     [,1] [,2] [,3] [,4]
[1,]   25    2    3    4
[2,]    5   25    7    8
[3,]    9   10   25   12
[4,]   13   14   15   25
```

Inversa

```
solve(mar)
```

```
              [,1]         [,2]         [,3]        [,4]
[1,]  0.0437911566 -0.001086095 -0.006589955 -0.01820922
[2,]  0.0007632019  0.049696187 -0.009275839 -0.02266123
[3,] -0.0015704347 -0.006105615  0.057548361 -0.03029325
[4,] -0.0064970010 -0.012798309 -0.023600552  0.06470582
```

Multiplicação matricial

```
M <- matrix(1:12,3,4); V <- matrix(13:24, 4, 3); M%*%V
```

```
     [,1] [,2] [,3]
[1,]  334  422  510
[2,]  392  496  600
[3,]  450  570  690
```

Nomeando matrizes

```
alunos = matrix(c(1:10, 4:8), ncol = 3);alunos
```

```
     [,1] [,2] [,3]
[1,]    1    6    4
[2,]    2    7    5
[3,]    3    8    6
[4,]    4    9    7
[5,]    5   10    8
```

Alterando o nome das linhas e colunas

```
colnames(alunos) = c("numero", "prova1", "prova2");
rownames(alunos) = c("Ari", "Bruno", "Cristina", "David", "Zulmira")
alunos
```

```
         numero prova1 prova2
Ari           1      6      4
Bruno         2      7      5
Cristina      3      8      6
David         4      9      7
Zulmira       5     10      8
```

Operações usando os novos nomes

```
mean(alunos[ ,"prova2"])
```

```
[1] 6
```

Listas

Coleção ordenada de objetos:

```
l = list(nome="Anderson", mulher="Renata", num.animais=2,
         idade.animais=c(0,1))
```

Veja:

```
l[[2]]; l[[4]][2]; l$mulher; l$idade.animais[1]
```

```
[1] "Renata"
```

```
[1] 1
```

```
[1] "Renata"
```

```
[1] 0
```

Diferença entre l[1] e l[[1]]

```
l$m
```

```
[1] "Renata"
```

```
k <- "nome"; l[[k]]
```

```
[1] "Anderson"
```

```
l[1]; l[[1]]
```

```
$nome
[1] "Anderson"
```

```
[1] "Anderson"
```

Data Frames

São tipos restritos de listas:

Vetores e matrizes com mesmo tamanho

```
dframe = data.frame(z1 = 1:10, z2 = 1:5); dframe
```

```
   z1 z2
1   1  1
2   2  2
3   3  3
4   4  4
5   5  5
6   6  1
7   7  2
8   8  3
```

```
9   9  4
10 10  5
```

Data frame pode virar matriz:

```
A <- data.matrix(dframe); A
```

```
      z1 z2
 [1,]  1  1
 [2,]  2  2
 [3,]  3  3
 [4,]  4  4
 [5,]  5  5
 [6,]  6  1
 [7,]  7  2
 [8,]  8  3
 [9,]  9  4
[10,] 10  5
```