Universidade Federal da Bahia
Instituto Multidisciplinar em Saúde
Curso de extensão: Introdução ao uso do programa R
Prof. Leandro Martins de Freitas e Profª Leila Cruz

Introdução ao uso do programa R

Dia 1: Fundamentos da Linguagem R

Olá, neste primeiro dia você aprenderá os primeiros passos para entrar nesse novo mundo da Linguagem R. Um esclarecimento inicial que pode facilitar a sua experiência: como o nome já indica, a linguagem R é uma ***linguagem***. Aprender a usá-la é como aprender um novo idioma. É preciso perseverar, praticar sempre para ganhar fluência e ter paciência porque a curva de aprendizagem é uma exponencial positiva: começa devagar, mas acelera e aí quem sabe onde você vai parar?

Hoje você verá:

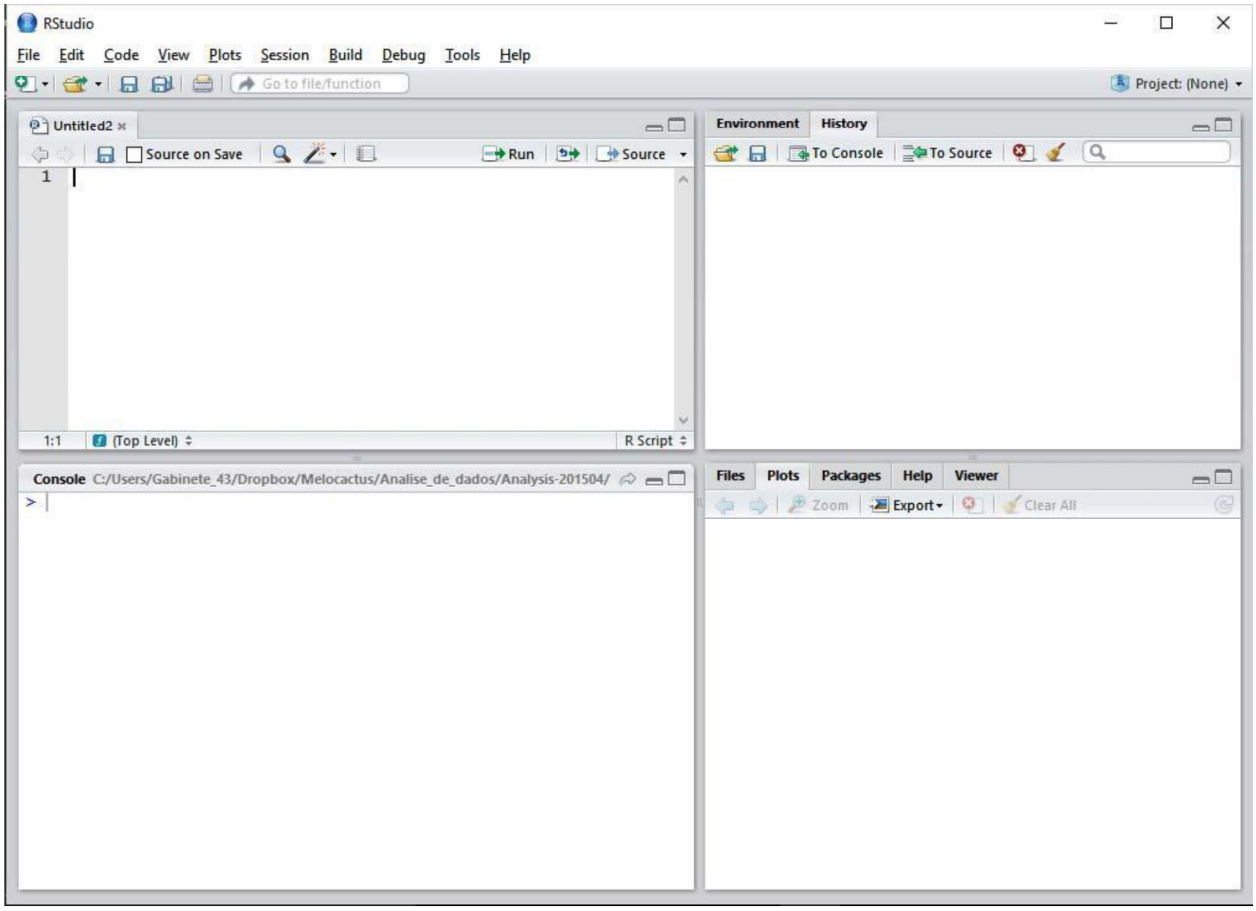
1. Como iniciar uma sessão de trabalho no RStudio
2. Variáveis e tipos de dados
3. Objetos do R: Variáveis, vetores, matrizes, data frames e listas
4. Gerenciando sua sessão: salvando seu espaço de trabalho (*workspace)*
5. Obtendo ajuda

1. RStudio

O RStudio é uma plataforma de programação no R mais "amigável" para o usuário. Sua área de trabalho está dividida em quadrantes, para compartimentalizar as informações e ações mostrando-as do modo mais completo possível. Você pode fazer o download do arquivo de instalação do RStudio aqui: https://www.rstudio.com/products/RStudio/

A tela inicial do RStudio é dividida em quadrantes. Nos quadrantes do lado esquerdo estão sua área de inserção de comandos, a janela "Script"; aqui você pode copiar, colar e editar textos e comandos, antes de fazer com que sejam executados pelo R. Depois de "entrar" (digitar) um comando, ele só será executado quando o usuário digitar "Ctrl+R": isso faz com que aquela linha de código digitada seja "lida" e computada pelo R, que retornará uma resposta. Essa leitura e computação é mostrada no painel esquerdo inferior, o "Console". Aqui será mostrando o código que foi lido, na cor azul, e o resultado computado correspondente, na cor preta, logo abaixo. O Console corresponde à janela que seria aberta se você estivesse usando a interface do R que foi descarregada da *CRAN - Comprehensive R Archive Network*.

Nos quadrantes do lado direito estão painéis de visualização de muitas informações sobre os códigos que estão sendo inseridos na sessão de trabalho. No quadrante superior direito temos duas abas, a “Workspace” que mostra os “objetos”[1] carregados até o momento e a “History” que mostra os comandos utilizados até então. No quadrante inferior direito temos cinco abas. Em “Files” temos as pastas de seu diretório “Documentos” do Windows. Na aba “Plots” podemos visualizar os gráficos que foram gerados nessa sessão. A visualização dos pacotes instalados e carregados na sessão na aba “Packages”. Na aba “Help” são mostradas as páginas de ajuda oficiais que estão armazenadas na CRAN. Por fim, na aba “Viewer” podem ser mostrados conteúdos disponíveis *on line*.



Veja abaixo mais informações sobre como utilizar os painéis e sobre o “fluxo de trabalho” durante uma sessão no RStudio.

1. Quadrantes

[1] O R é uma linguagem de programação “orientada a objetos”. Isso significa que cada linha de comando deve operar sobre um “objeto”: uma variável, um vetor, um data frame, uma função. Estes objetos têm as propriedades particulares de sua classe.

1.1 Script

Escrevemos os comandos em um script que será executado pelo R. O script é um conjunto de comandos que realizará uma tarefa (análise).

O scritpt pode ser executado de uma unica vez, ou seja, executando todas as linhas. Podemos também executar uma linha por vez ou um conjunto de linhas selecionadas, isso traz mais liberdade para corrigir erros e executar comandos antes de chegar em partes mais complexas do script.

Devemos usar comentários dentro do script para lembrar ou explicar o que está sendo executado. Comentários tornam o script mais profissional porque permitem que outra pessoas entendam sua forma de analisar e programar.

Para inserir comentário em um script devemos usar o simbolo #. O que for adicionado depois do # não será lido pelo programa R.

Exemplo de comentário:

```
#Autor: Leandro Martins de Freitas
#data: 30122015
#primeiro script produzido no Curso Introdução ao uso do programa R - 2015.
#esse script estima a média
```

1.2 Linha de comando

Comandos são digitados diretamente na linha de comando, indicada por >. Após digitar o comando para executar deve pressionar a tecla Enter. O resultado aparece no visor abaixo. Não armazenado na memória. Não pode ser recuperado.

Algumas operações não retornam mensagem, portanto somente o sinal > ficará disponível para um novo comando. Outras operações podem demorar muito tempo para serem processadas. Devemos aguardar até o simbolo > ficar disponível novamente. Caso o usuário deseje cancelar um comando que está sendo executado, devemos pressionar a tecla Esc.

2. Variáveis e tipos de dados

Os tipos de variáveis básicas do R são: Character, Number e Logical

2.1 Character

As variáveis do tipo Character armazenam informações tipo letras e palavras (*strings*). As variávéis do tipo Character podem armazenar uma única letra ou conjunto de letras seguidas (com ou sem espaço entre diferentes strings). Também podemos armazenar número como Character, entretanto não poderemos fazer operações matemática com esses números[2].

---

[2] Podemos converter uma variável tipo Character que armazene informação numérica para uma variável numérica. Para isso devemos usar a função as.numeric(). Exemplo:

Escreva em um script com os seguintes valores tipo Character e execute linha por linha. Você deve observar quais mensagens são retornadas pelo R.

```
>"Hello world"
>"A"
>"Emerita brasiliensis"
>"Emerita brasiliensis "
>"Panthera onca"
>"Panthera onça"
>"Emerita brasiliensis" == "Emerita brasiliensis "
>"Panthera onca" == "Panthera onça"
```

2.1.1 Não devem ser utilizados valores do tipo Character da seguinte forma.

Veja a mensagem retornada pelo R para os casos abaixo:

```
>"Ananas comosus"  > "Emerita brasiliensis"
>"ABC" < "XYZ"
>"Emerita brasiliensis" + "Ananas comosus"
>A
>Panthera onca
```

2.1.2 Atribuição de valores em variáveis.

Podemos atribuir os valores tipo Character para uma variável[3]. Os valores são atribuidos as variáveis usando o <- ou =. Nesse documento usaremos <- para fazer atribuição de valores.

```
>a<-"Hello world"
>especie1 <- "Panthera onca"
>a
>especie1
```

```
>as.numeric("10")
```

[3] Quando guardamos valores em variáveis estamos reservando um espaço na memória do computador para guardar essa informação por um tempo. O valor armazenado pode ser recuperado.

2.2 Numeric

As variáveis do tipo Numeric armazenam informações de numérica. As variávéis do tipo Numeric podem armazenar números positivos quanto negativos do tipo inteiro ou decimal.

O R realiza operações matemáticas usuais. Em uma expressão com múltiplas operações, a ordem de computação dos resultados é: exponenciação → multiplicação ou divisão → soma ou subtração. A ordem de computação das operações depende também da "comlartimentalização" da expressão: todas as operações dentro de parênteses são realizadas antes que as que estiverem fora, sempre seguindo a ordem de prioridade anterior.

Escreva em um script com os seguintes valores tipo Numeric e execute linha por linha. Você deve observar quais mensagens são retornadas pelo R.

```
>2
>3.6
>-30.2
>-10.2567
>2*3
>2+3*5
>(2+3)*5
>catetoA<-2
>catetoB<-3
>catetoA+catetoB
>quadrado.hipotenusa<-catetoA^2+catetoB^2
>sqrt(quadrado.hipotenusa)
>1e+10 # significa 10**10
>1.267651e+30 # significa 2**100
```

2.2.1 Não devem ser utilizados valores do tipo Numeric da seguinte forma.

Veja a mensagem retornada pelo R para os casos abaixo:

```
>23,7
>1 0
>-30. 208
```

2.2.2 Atribuição de valores em variáveis.

Podemos atribuir os valores tipo Numeric para uma variável.

```
>numero1 <- 10
>numero_neg <- -107
```

```
>media <- 90.87
```

2.2.3 Operações

Adição (+)

```
>120 +100
>-34+50
```

Subtração (-)

```
>50-39
>-10-45
```

Multiplicação (*)

```
>9*9
>2*45
```

Divisão (/)

```
>10/2
>67/21
```

Potenciação (^ ou **)

```
>2^3
>45^4
```

Radiciação (sqrt)

```
>sqrt(100)
>sqrt(3)
```

Prioridades

Devido a prioridade nas operações matemáticas devemos separadas por () partes das equações com ().

Exercícios

2.3 Logical

As variáveis do tipo Logical armazenam somente dois tipos de informação, `TRUE` ou `FALSE`. Essa informação também é aceita pelo programa de forma simplificada como `T` ou `F`.

Escreva em um script com os seguintes valores tipo Logical e execute linha por linha. Você deve observar quais mensagens são retornadas pelo R.

```
>TRUE
>FALSE
#ou
```

```
>T
>F
```

2.3.1 Nao deve ser utilizado valores do tipo Logic da seguinte forma.

```
>True
>False
#Qual a diferença para TRUE e "TRUE", ou FALSE e "FALSE"?
>"FALSE"
>"TRUE"
```

2.3.2 Atribuição de valores em variáveis.

```
>valor <- T
>compara <- FALSE
>compara2 <- 10 < 20
>resultado <- 100-20*2 > 50
>compara
>compara2
>resultado
```

2.3.3 Nome de variáveis

Procure colocar nome simples e lógico: indicando o que a variável contem. Nomes como A, X var não indica o que a variável armazena. Tanto o autor como outros usuário podem ficar confusos com esses nomes.

Regras
Sempre começar com letra (nome, idade, tamanho)
Pode usar número (mediaPop1, mediaPop2)
Pode usar _ ou . (Pop.1; Pop_2)
Case sensitive (nome; Nome; NOME)
Não usar c, q, t, C, D, F, I, T, diff, df, pt, pi. Essas já são funções ou variáveis definidas pelo R.

2.3.4 Variáveis já criadas pelo usuário

Para consultar quais variáveis já foram criadas devemos usar a função `ls()`.

```
>ls()
```

2.3.5 Tipo de variáveis - classe das variáveis

Devemos consultar qual a classe das variáveis para poder usar análises corretas e evitar mensagens de erro.

Para saber qual a classe de uma variável devemos usar a função `class()`

```
>class(dados)
>class(lista)
>class(conjunto)
```

Consulte o tipo class() de cada variável criada até agora nesse tutorial.

2.4 Apagando variáveis

Podemos liberar memória do computador apagando variáveis que não iremos usar mais, para isso usamos a função `rm()`.

Essa é uma função perigosa porque se usada incorretamente irá apagar todas as sua variáveis já criadas.

```
>rm(conjunto) #apaga a variável conjunto
>rm(list=ls()) # apaga todas as variáveis do ambiente
```

3. Vector

Vetores são variáveis que armazenam mais de um elemento. Para criar um vetor devemos usar a função `c()`, combina valores em um vetor. Os elementos do vetor podem ser acessados usando o índice da posição onde o elemento está no vetor.

Todos os elementos dentro do vetor devem ser do mesmo tipo (Character, Numeric, ou Logic).

Devemos inserir os elementos no vetor usando "", separados por ",". Os elementos do vetor podem ser todos únicos ou apresentar elementos repetidos. O indice dos elementos irá ser o identificador único do elemento, permitindo acessar cada elementos de forma específica ser for desejado.

Vetor tipo character

"A"	"B"	"A"	"C"	"D"

Índice do vetor

1	2	3	4	5

Escreva em um script criando os vetores abaixo com os seguintes valores e execute linha por linha. Você deve observar quais mensagens são retornadas pelo R.

3.1 Character

```
>b <- c("B", "Emerita brasiliensis", "Emerita brasiliensis ",
"Panthera onca" , "Panthera onça")
>cidades <- c("Salvador", "Vitória da Conquista","Belo
Horizonte","Manaus", "Salvador")
```

3.2 Numeric

```
>conjunto <- c(1, 10, 23, 59, 1, 1:10)
>elementos <- c(1, 1, 2, 3, 5, 6, 4)
>a <- c(1,2,5.3,6,-2,4) # numeric vector
```

3.3 Logical

```
> resultado <- c(T, F)
> resultado2 <- c(TRUE, FALSE, TRUE, TRUE, FALSE)
>c <- c(TRUE,TRUE,TRUE,FALSE,TRUE,FALSE)
```

```
>resultado <- c(True, False)
>c(1,2,3,10,5,25,43) > 10
```

3.4 Factor

Factor é um vetor que separa os elementos em categorias, designadas por valores inteiros. O Factor usa um número limitado de diferentes valores e codifica esse elementos com números inteiros. Algumas funções esperam um vetor tipo Factor, caso contrário elas retornam erro.

```
>conjunto <-
c("A","B","B","A","C","B","M","B","A","A","C","A","B","A","B","A","A",
"A","C")
>conjunto.f <- factor(conjunto)
>conjunto
>conjunto.f

#Exemplo da diferença entre character e factor
>plot(conjunto)
>plot(conjunto.f)
```

3.5 Casos especiais

```
>c(1,2,3, x)
>c(1,2,3, "a")
```

3.4 Símbolos especiais

NA/NaN simbolos que podem estar dentro do vetor
NA = missing value
NaN = Not a Number

```
>novoVetor <- c(NA, 1, 3, 6)
>novoVetor <- sqrt(novoVetor)

>novoVetor2 <- c(NaN, 100, 49, -1)
>novoVetor2 <- sqrt(novoVetor2)
```

A presença de NA ou NaN no vetor não muda o seu tipo. Nos dois exemplos acima os vetores continuam sendo tipo Numeric.

Exercício

1. Em Ecologia, muitos trabalhos buscam calcular Índices de Diversidade de comunidades com várias especies diferentes. Estes índices consideram tanto o número de espécies diferentes encontradas em uma amostra quanto o número de individuos de cada espécie. Um dos mais utilizados é o Índice de Shannon-Wiener (**H'**), que tem a fórmula

$$\mathbf{H' = -\sum [p_i \cdot \ln(p_i)]}$$

sendo que:

p_i é a proporção de indivíduos encontrados na espécie i (frequência relativa)

$\ln(p_i)$ é o logaritmo natural de p_i.

a. Calcule **H'** para a seguinte comunidade:

Espécie	Número de indivíduos (N)
A	16
B	9
C	121
D	126
E	15
F	2

b. Calcule a média e o desvio padrão das frequências de indivíduos desta comunidade

c. Qual o tipo de dado dos vetores de todas as espécies e do número de indivíduos?

d. Compare as frequências de indivíduos das espécies G (N=88) e H (N=9) coletados em uma nova amostragem em relação às frequências as espécies na tabela. Que espécies da primeira coleta têm frequências maiores e menores que G e H?

e. Qual a diferença entre as frequências de cada espécie e a média?

Operações com vetores

Combining Vectors
Vector Arithmetics
Vector Index
Numeric Index Vector
Logical Index Vector

Named Vector Members

4. Matrix

O tipo Matrix é uma variável que armazena valores do mesmo tipo, assim como os vetores. A estrutura de dados é “bidimensional”. Uma matriz mxn possui m x n elementos.

Criando uma matriz no R

```
>matriz <- matrix(c("Pedro da Silva", "José de Oliveira", "Belo
Horizonte", "Vitória da Conquista"), nrow = 2, ncol =2, dimnames =
list(c("linha1", "linha2"), c("C.1", "C.2")))

>matriz
>elementos <- c(1,10,32,43,1,2,7,2,6,1)
>nova_matriz <- matrix(c(elementos, elementos*2), ncol=2)
>nova_matriz
```

Acessando os elementos da matriz

Para acessar os elementos da matriz devemos usar indice linha e indice coluna.

	1	2	3	4
1	1	70	90	64
2	45	70	62	37
3	12	71	100	0

Os números da matriz acima destacados em vermelho são índices das linhas, enquanto números da matriz acima destacados em azul são índices das colunas. Devemos usar [ ] para indicar quais elementos da matriz vamos usar. O primeiro índice é sempre a linha e o segundo é sempre a coluna. [linha, coluna].

```
#iremos acessar a linha 1 e a coluna 2
>matriz[1,2]
#iremos acessar a linha 2 e a coluna 2
>matriz[2,2]
```

Podemos recuperar quais são o nomes usados como linha e colunas usando as funções `rownames()` e `colnames()`.

```
>colnames(matriz)
>rownames(matriz)
```

Essas funções também podem ser usadas para mudar os nomes de linhas e colunas.

```
>colnames(matriz) <- c("coluna1", "coluna2")
>rownames(matriz) <- c("nome_nome1", "novo_nome2")
>colnames(matriz)
>rownames(matriz)
```

Outra forma de recuperar elemento de uma matriz é indicar somente qual é a linha ou coluna. Isso irá recuperar todos elementos da linha ou coluna.

Quais elementos são recuperados com os comandos abaixo?

```
>nova_matriz[,2]

>nova_matriz[3,]
```

5. Data-frame

O tipo data.frame é uma variável que armazena valores que podem ser de tipos diferentes (numeric, character, logic, factor). Entretanto cada coluna armazena somente valores do mesmo tipo, assim como os vetores. A estrutura de dados é “bidimensional”.

	Nome	Idade	família_real	variavel1
ID1	“Jose da Silva”	45	FALSE	A
ID2	“Pedro de Bragança e Bourbon”	35	TRUE	A
ID3	“Pedro de Oliveira”	60	FALSE	B

ID3	“ Isabel de Bragança”	25	TRUE	A

```
>nome <- c("Pedro da Silva", "José de Oliveira")
>idade <- c(41, 18)
>cidade <- c("Belo Horizonte", "Vitória da Conquista")

>dados <- data.frame(nome, idade, cidade)
>dados
```

Da mesma forma que a matriz, podemos acessar os valores do dafa frame usando indices de linha e coluna.

```
>dados[1,2]
>dados[3,4]
```

Outra forma de recuperar elemento de um data frame é indicar somente qual é a linha ou coluna. Isso irá recuperar todos elementos da linha ou coluna.

```
>dados[,2] #ou dados$idade
>dados[,3] #ou dados$cidade
```

Podemos também mudar nomes de linhas e colunas usando as funções `rownames()` e `colnames()`.

```
>colnames(dados) <- c("Nomes2","Idades2","Cidade2")
```

As funções `attach()` e `deattch()` são para facilitar o trabalho com data frame. A função `attach()` faz com que cada coluna do data frame se torne uma variavél global. Dessa forma podemos usar somente o nome da coluna para fazer uma operação.

```
>attach(dados)
>Nomes2
>Idades2
>Cidade2
```

Para remover as colunas do data frame das variáveis globais usamos a função `deattch()`.

```
>detach(dados)
```

6. List

A lista é um objeto e pode apresentar diferentes características. As características podem ser de diferentes tipos e diferentes comprimentos (número de elementos).

Iremos agora criar uma lista que contém 5 características.
Característica 1 contém a letra “A”
Característica 2 contém o número 37
Característica 3 vetor com 10 elementos numéricos
Característica 4 vetor com 5 elementos logicos
Característica 5 vetor com 3 elementos character

```
>lista<-list()
>lista[[1]]<-"A"
>lista[[2]]<-37
>lista[[3]]<-c(30:21)
>lista[[4]]<-c("TRUE","FALSE","TRUE","FALSE","FALSE")
>lista[[5]]<-c("red","blue","red")
>lista
3.2 acessando elementos da lista
> lista[[1]] #retorna todos os elementos dessa caracteristica 1
> lista[[2]]  #retorna todos os elementos dessa caracteristica 2
> lista[[3]]  #retorna todos os elementos dessa caracteristica 3
>lista[[3]][4] #retorna somente o elemento 4 da caracteristica 3
```

Exercício

02. O conjunto de dados “iris” contém medidas biométricas da corola de 150 indivíduos de três diferentes espécies de flores do gênero *Iris*, *Iris setosa*, *Iris versicolor* e *Iris virginica*.
    a. Que variáveis estão armazenadas nesse conjunto de dados? Quantas observações dessas variáveis para cada espécie ele contém?
    b. Calcule a média e desvio padrão dessas variáveis para cada espécie de *Iris*.
    c. Construa uma tabela com valores mínimos, máximos, média e desvio parão dessas variáveis para as três *Iris*, nomeando as colunas e linhas apropriadamente.

7. Instalando e carregando pacotes

Podemos aumentar o número de funções do R carregando pacotes. Os pacotes são desenvolvidos por usuários do R e são lançados para que outros usuários possam usar das análises desenvolvidas. Para instalar um pacote devemos usar a função `install.packages()`.

```
>install.packages("cluster", dependencies=T)
```

Devemos carregar o pacote para que as funções fiquem disponíveis. Para isso, usamos a função `library()` para carregar um pacote.

```
>library("cluster")
```

Podemos agora usar funções espécificas do pacote cluster

```
#criando dados para testar na função pam() do pacote cluster.

>x <- rbind(cbind(rnorm(10,0,0.5), rnorm(10,0,0.5)),
cbind(rnorm(15,5,0.5), rnorm(15,5,0.5)))
```

Função `pam()` faz particionamento dos dados em grupos. Devemos passar como parâmetros os dados e o número de grupos que a função deve dividir os dados. A função tentará fazer a melhor divisão dos dados no número de grupos desejados.

```
>grupos <- pam(x,2) # x é o conjunto de dados, 2 é o número de grupos
```

A função `clusplot()` faz um gráfico representando a separação dos dados nos grupos definidos pela função `pam()`.

```
>clusplot(grupos)
```

8. Help

Sempre precisamos pedir ajuda para montar testes, scripts, e saber como executar os comandos no R. Como cada pacote tem muitas funções, com muitos parâmetros é necessário com frequência consultar a ajuda para usar a sintaxe correta.

Podemos usar o help do R de várias formas.

Função `apropos()` retorna todas as funções do R e pacotes que contém o termo pesquisado. Ideal para uma busca ampla sobre o termo e não é sensível a maiuscula ou minuscula.

```
>apropos("mean")
```

Usando a função `help()` e o nome da função que deseja pesquisar. Para usar essa forma devemos saber exatamente o nome da função no R.

```
>help(mean)
>help(Mean) #erro porque não existe função Mean começando com
maiuscula.
```

Outra forma de buscar por ajuda é usar o simbolo de interrogação.

```
> ?mean
> ?Mean  #erro porque não existe função Mean começando com maiuscula.
```

Para pesquisar ajuda para caracteres especiais devemos colocar entre aspas, trasnformando ele em tipo Character. O mesmo é válido para palavras que representam alguma operação na linguagem de programação R, como *for*, *if*, *ifelse*.

```
>help("[") #pesquisa como usar o simbolo [
>help([) #erro porque o simbolo [ não foi transformado em carácter
```

Podemos navegar pela ajuda do R usando a função `help.start()`. Isso irá abrir uma página local HTML com links para diversas partes da ajuda do R.

```
>help.start()
```

Navegue pela página aberta pela função `help.start()` e veja quais informações podem ser encontradas.

Ajuda quando não sabemos o nome exato da função deve ser realizada usando a pesquisa com `??`

```
>??Mean
```

Quando nossa busca não retorna nenhum resultado ou o resultado não atende a nossa demanda devemos buscar em outro lugar.

Exemplo

```
>??ecology
```

Próximo passo pesquisar nos pacotes do R, se tem algum pacote lançado que contém o termo da busca. O CRAN - *The Comprehensive R Archive Network* (https://cran.r-project.org/) armazena pacotes que podem ser instalados no R.

Vamos usar o pacote `sos` para fazer a busca por termos encontrados em outros pacotes.

Primeiro devemos instalar o pacote `sos` usando a função `install.packages()`, recomendamos sempre usar o parâmetro `dependencies=T`. Após a função ser instalada devemos carregar quando formos usar uma função desse pacote. Para carregar um pacote devemos usar a função `library()`.

```
>install.packages("sos", dependencies=T)
#carregando a função sos
>library("sos")
#usando a função findFn() para buscar o termo ecology
#para usar essa função é necessário conexão com a internet.
findFn("ecology")
```