BIE5782

Aula 9 e 10

NOÇÕES DE PROGRAMAÇÃO

Mensagem da Aula

ANIMAÇÃO

Primeiros passos no R!

Depois de fazer a primeira Função no R!

editoR de texto

NotePad++: free, precisa do NpptoR
TinnR: This is not a notepad, brasuca…

ESS: *Emacs Speaks Statistics* (ESS), várias plataformas (veja doc Musgo!)
JGR: java GUI to R pacote CRAN
Rstudio: roda em Java independe do OS
----------Linux----------
RKward: GUI e editor
Kate: editor integrado ao console
Gedit: plugin R integration
Emacs: ESS
Dicas&Didáticas: Labtrop

Editor no R

Exemplo: Rgedit, JGR, RStudio:

simula.r

function
Programação

function()

conjunto de comandos concatenados, desempenhando um fim!

TAREFA!

LINHA DE MONTAGEM conduzida por um algorítmo!

TODO USUÁRIO É UM PROGRAMADOR!!!

Algorítmo

"..é uma sequência não ambígua de instruções que é executada até que determinada condição se verifique. Mais especificamente, em matemática, constitui o conjunto de processos (e símbolos que os representam) para efetuar um cálculo."

Algorítmo

"..é uma sequência não ambígua de instruções que é executada até que determinada condição se verifique. Mais especificamente, em matemática, constitui o conjunto de processos (e símbolos que os representam) para efetuar um cálculo."

Algorítmo

SEQUÊNCIA DE INSTRUÇÕES PARA REALIZAR UMA TAREFA

METÁFORA DO COZINHEIRO
Como fazer um Petit Gâteau?
- objetos (ovo, manteiga, farinha, chocolate, panelas, forno)
- "Funções" que manipulam os objetos
- instruções sequenciais

Receita ~ Algoritmo do Cozinheiro

FUNÇÕES

OBJETO DA CLASSE 'FUNCTION'

NOME (não trivial)

1. **tarefa realizada: summary()**
2. **revela objetivo: plot()**
3. **acrônimo: lm(); dp()**
4. **memorável sunflowerplot()**
5. **não usual!: caso do pi**

FUNÇÕES

ARGUMENTOS

objetos e parâmetros necessários para executar a tarefa (cuidado com o padrão!)

```
sum(x, na.rm = FALSE)

read.table(file, header = FALSE, sep = "", quote
    = "\"'", dec = ".", row.names, col.names,
as.is = !stringsAsFactors, na.strings = "NA",
colClasses = NA, nrows = -1, skip = 0,
check.names = TRUE, fill = !blank.lines.skip,
strip.white = FALSE, blank.lines.skip = TRUE,
comment.char = "#", allowEscapes = FALSE,  flush
= FALSE, stringsAsFactors =default
.stringsAsFactors(),encoding = "unknown")
```

function(args){comandos}

Estrutura Básica

```
minha.funcao <- function(argumento1,
                    argumento2,. . .)
                {
                    comando 1
                    comando 2
                    comando 3 . . .
                    return()
                }
```

eda.shape()

Estrutura Básica

```
eda.shape <- function(x)
    {
    par(mfrow = c(2,2))
    hist(x)
    boxplot(x)
    iqd <- summary(x)[5] -  summary(x)[2]
    plot(density(x,width=2*iqd),xlab=
  + "x",ylab="",type="l")
    qqnorm(x)
    qqline(x)
    par(mfrow=c(1,1))
    }
```

Senta que la vem história!

Conquistando o MUNDO!

Código Livre.
Pode criar, modificar, copiar e distribuir.

General Public Licence
(GNU- GPL)

Free Software Foundation

eda.shape()

Estrutura Básica

```
eda.shape <- function(x)
   {
   par(mfrow = c(2,2))
   hist(x)
   boxplot(x)
   iqd <- summary(x)[5] -  summary(x)[2]
   plot(density(x,width=2*iqd),xlab=
   + "x",ylab="",type="l")
   qqnorm(x)
   qqline(x)
   par(mfrow=c(1,1))
   }
```

Primeiros passos no R!

Primeira Lição!!

O que acontece com quem se aventuram a programar uma função???

Depois de fazer a primeira Função no R!

O mais comum!

É uma versão piorada de uma função que já existe e vc. desconhecia!

function ()

Funções Simples

```
media <-function(x)
    {
    soma=sum(x)
    nobs=length(x)
    media=soma/nobs
    return(media)
    }
```

Vamos ao R!

If(){}; else{}

Funções Simples

```
If(condição)
    {
    comandos se condição = TRUE
    }
else
    {
    comandos se condição = FALSE
    }
```

function ()

Funções Simples

```
media<-function(x,rmNA=TRUE)
     {
     if(rmNA==TRUE)
          {
          dados=(na.omit(x))
          n.NA=length(x)-length(dados)
          cat("\t", n.NA," valores NA excluídos\n")
          }
          else
          {
          dados=x
          }
     soma=sum(dados)
     nobs=length(dados)
     media=soma/nobs
     return(media)
     }
```

Vamos ao R!

Conceito de dispersão

Biológia x Matemática

Índice de Dispersão
Razão: Variância/Média

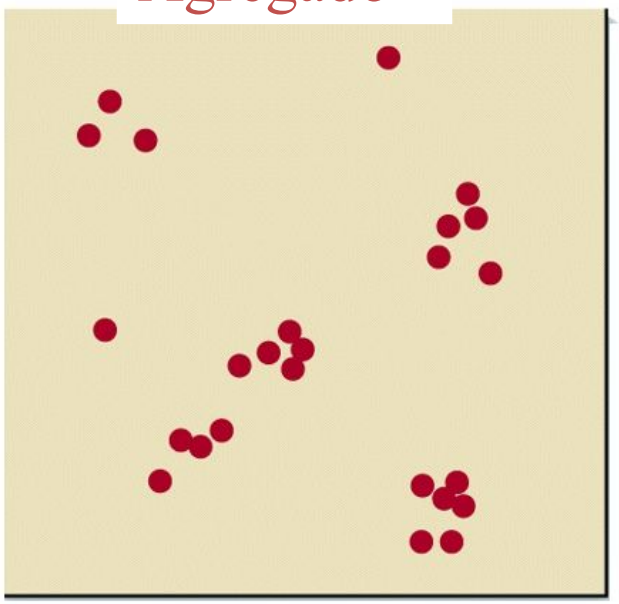
Uniforme: os valores observados estão próximos a média

Aleatório: os valores observados estão dispersos aleatoriamente ao redor da média

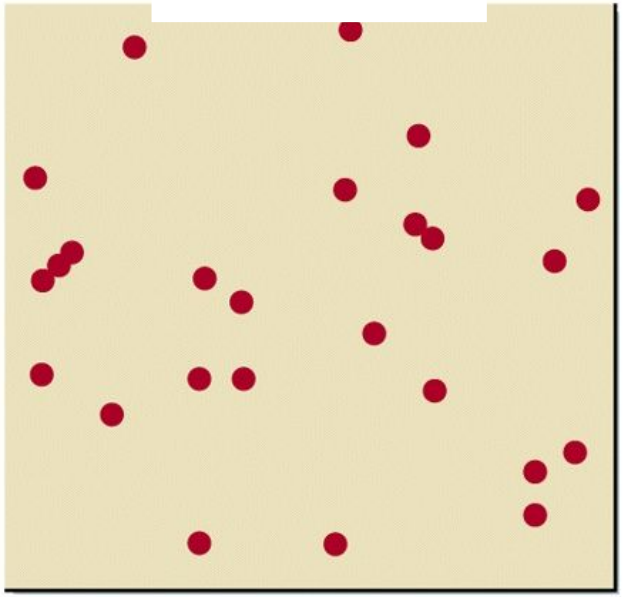
Agregado: os valores estão mais dispersos do que seria esperado ao acaso

Padrões de distribuição

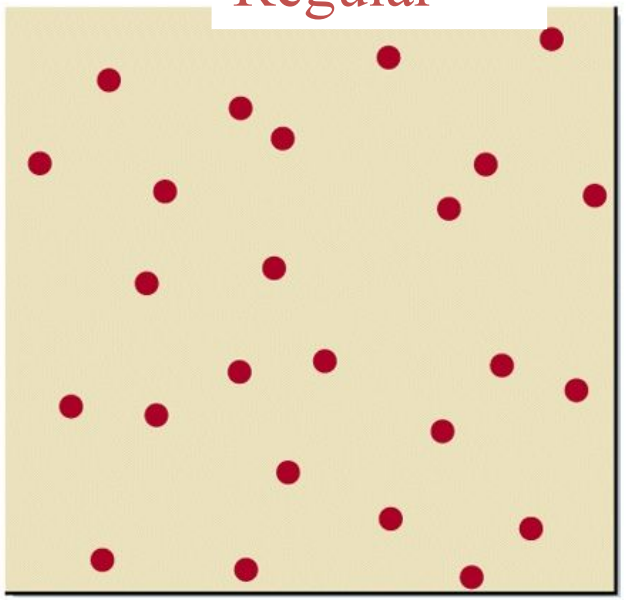
Agregado



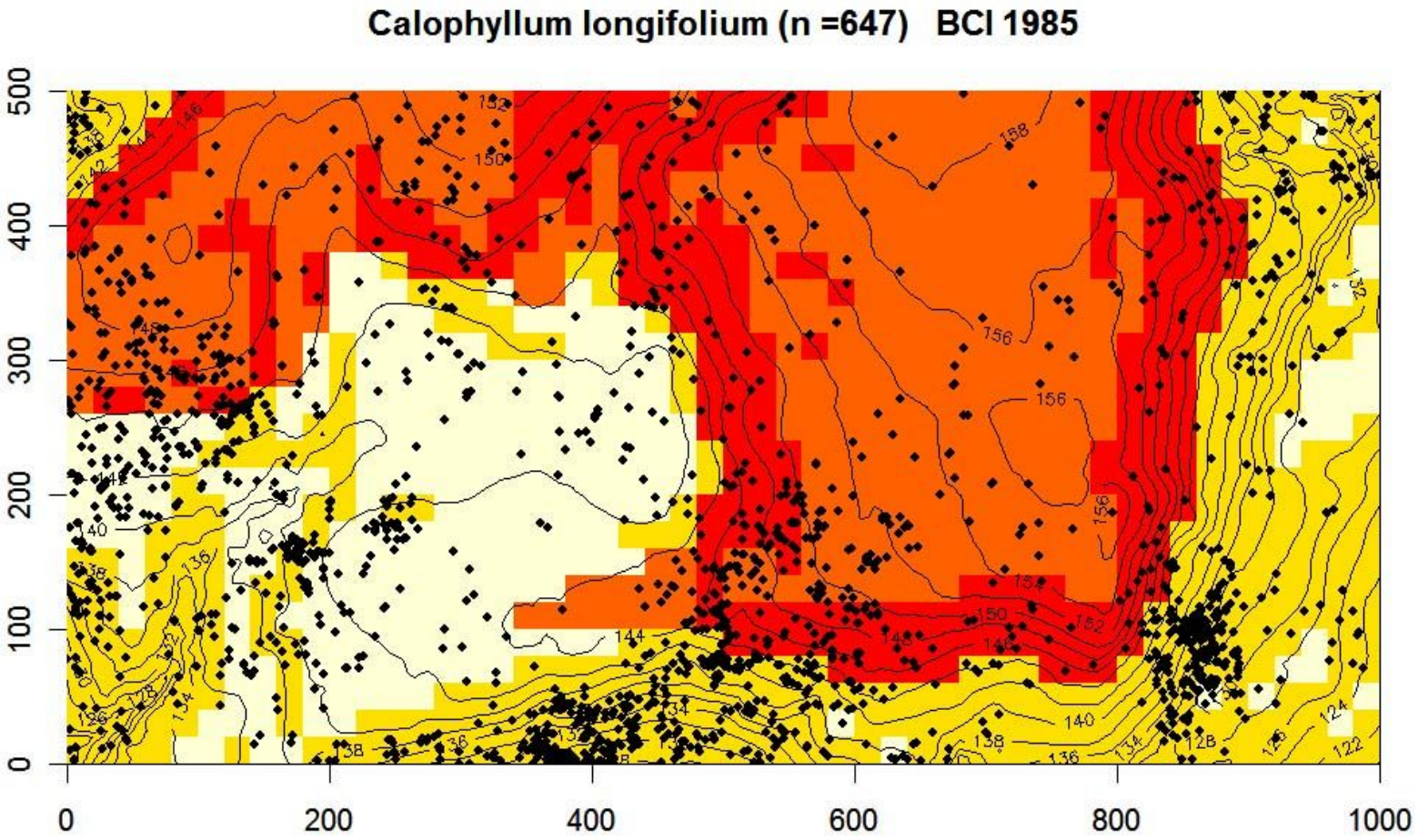
Aleatório



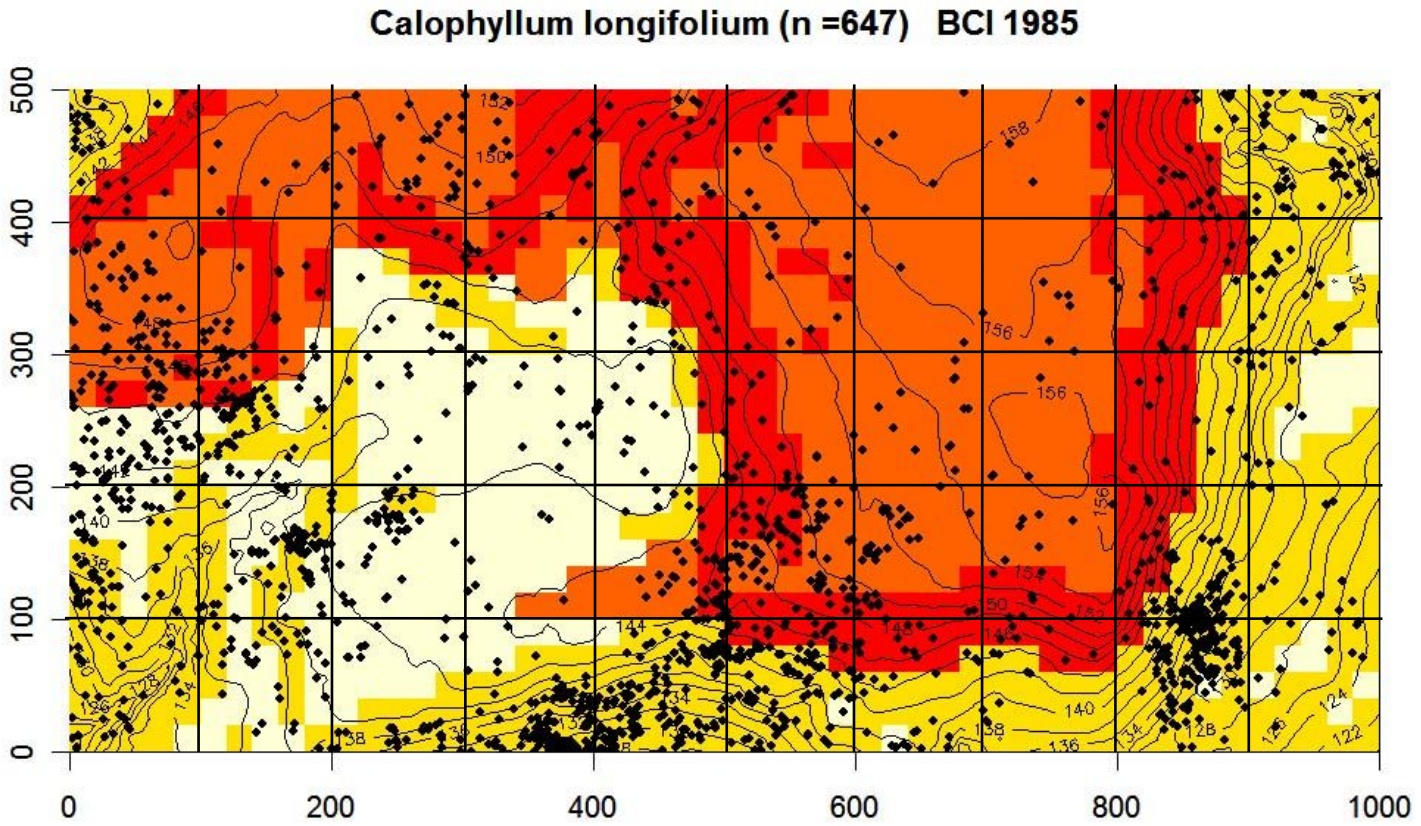
Regular



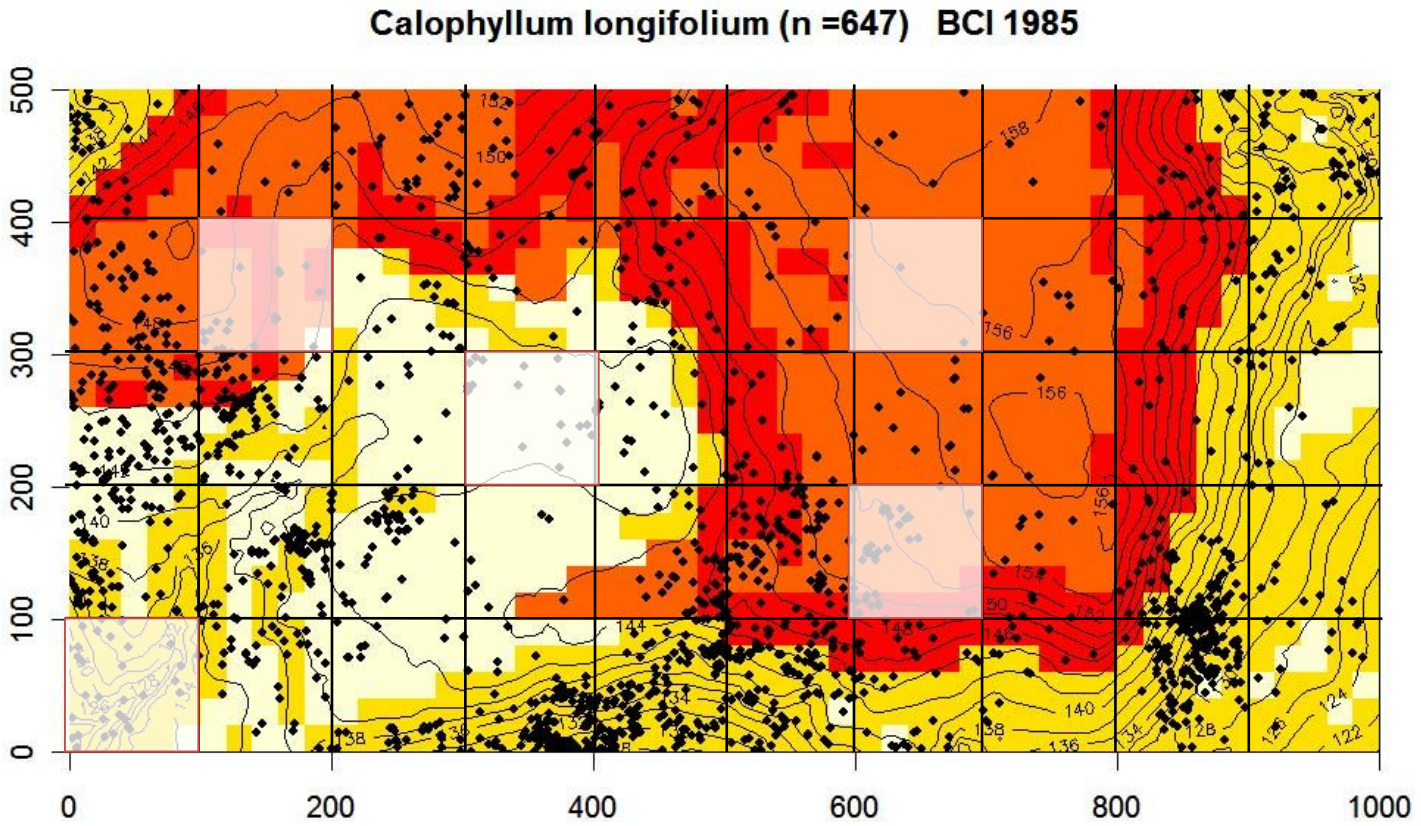
Distribuição Espacial



Distribuição Espacial

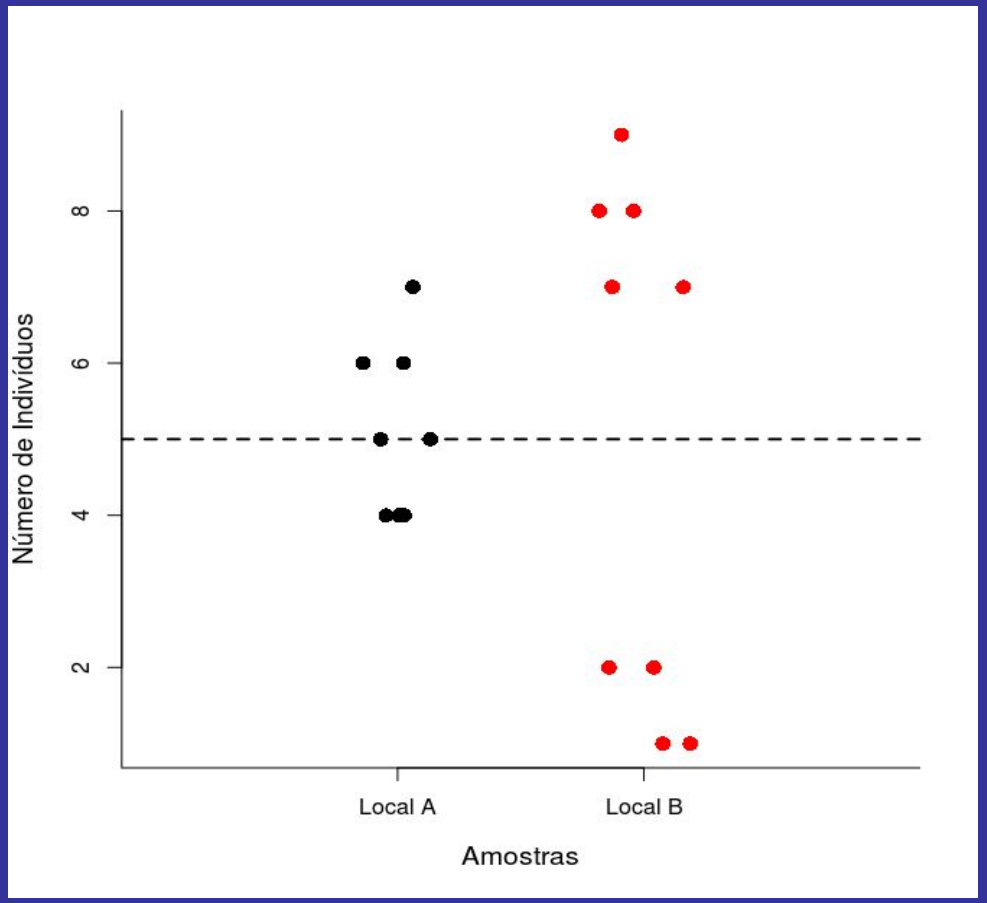


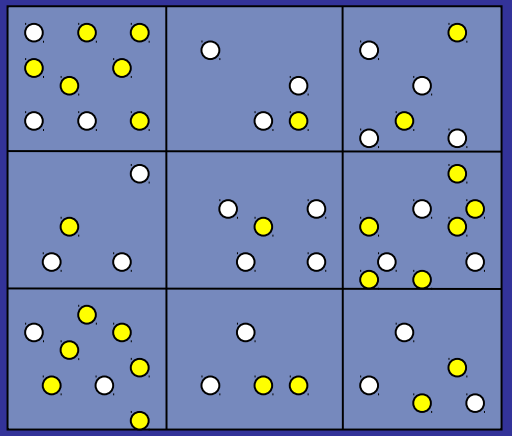
Prática – Sorteio



Descrição do Padrão Espacial

- Média: noção da tendência central
- Variância: noção da dispersão





ID.curso()
Funções Dispersão

```
ID.curso<-function(x)
    {
    id=variancia(x)/media(x)
    return(id)
    }
```

Vamos ao R!

If(){}; else{}

Funções Simples

```
If(condição)
    {
    comandos se condição = TRUE
    }
else
    {
    comandos se condição = FALSE
    }
```

teste.ID ()
Funções Dispersão

```
test.ID <- function(x, nsim=1000)
{
     ID.curso=function(x){var(x)/mean(x)}
     dados=na.omit(x)
     ndados=length(dados)
     med=mean(dados)
     id=var(dados)/med
     simula.aleat=rpois(length(x)*nsim, lambda=med)
     sim.dados=matrix(simula.aleat,ncol= ndados)
     sim.ID=apply(sim.dados,1,ID.curso)
     quant.ID=quantile(sim.ID, probs=c(0.025,0.975))
          if(id>=quant.ID[1] & id<=quant.ID[2])
           {
cat("\n\t distribuição aleatória para alfa=0.05
     \n\tID=",id,"\n")
           }

...
```

teste.ID ()
Funções Dispersão

```
...
if(id < quant.ID[1])
     {
     cat("\n\t distribuição uniforme, p<0.025 \n\t ID=
          +",id,"\n")
     }
     if(id>quant.ID[2])
     {
     cat("\n\t distribuição agregado, p>0.975 \n\t ID=
      +",id,"\n")
     }
resulta=c(id,quant.ID)
names(resulta)<-c("Indice de Dispersão", "critico 0.025",
"critico 0.975")
return(resulta)
}
##########################################################
######
```

Vamos ao R!

for(){}

Funções com ciclos

dados = matrix (20 spp , 4 parcelas)

```
x1=rpois(20,5)
x2=rpois(20,2)
x3=rpois(20,3)
x4=rpois(20,1)
sp.oc=matrix(c(x1,x2,x3,x4), ncol=4)
colnames(sp.oc)<-c("plot A", "plot B",
          "plot C", "plot D")
rownames(sp.oc)<-paste("sp", c(1:20))
str(sp.oc)
dim(sp.oc)
head(sp.oc)
```

for(){}

Funções com ciclos

dados = matrix (spp , parcelas)

```
n.spp<-function(dados)
   {
   nplot=dim(dados)[2]
   resultados=rep(NA,nplot)
   names(resultados)<-paste("n.spp",c(1:nplot))
   dados[dados>0]=1
   for(i in 1:(dim(dados)[2]))
      {
      cont.sp=sum(dados[,i])
      resultados[i]=cont.sp
      }
   return(resultados)
   }
```

Vamos ao R!

debug() ; undebug ()

Programando

Função (desespero total!!!!)

debug(nome da função)

-roda a função para cada comando e pode mostrar as variáveis
-problema: toda vez que chamar a função ela estará em debug...
-para sair deve dar "Q" e depois:

undebug(função)

Dicas Funções

- Documente os passos, internamente, inclusive autoria e versão

- Explicite o formato do objeto de entrada

- Faça uma biblioteca de funções por aplicação (use o "save workspace")

- Prefira opções de argumento a nova funções, mas não exagere

- Compartimentalize problemas complexos em funções internas mais simples

-

Carregando Funções

Para rodar uma função no R:

```
source("C:\\Users\\Alexandre\\r\\aula\
\2009\\aulaProgramar\\eda.shape.R")
```

Para evitar erro de digitação o `choose.file()` ou o pelo menu do Rgui "file", "source R code"

"Copie" do editor de texto e "Cole" no console do R tb. funciona

Agora que fiz minha primeira funcão...