BIE5782

Aula 9 e 10

NOÇÕES DE PROGRAMAÇÃO

Mensagem da Aula

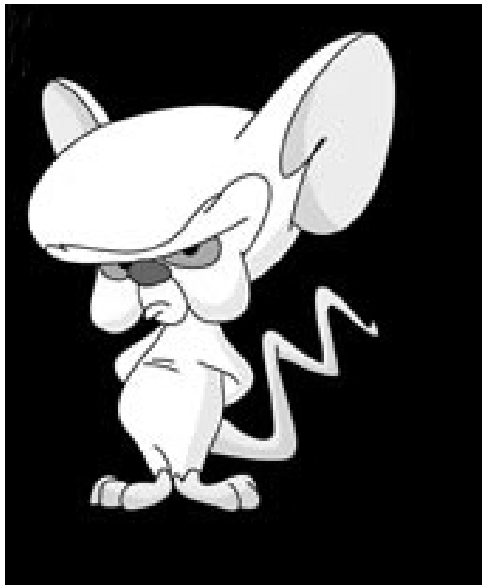
Aula 9 e 10

NOÇÕES DE PROGRAMAÇÃO

Animação!

Primeiros passos no R!

Depois de fazer a primeira Função no R!



editoR de texto

RWinEdt : pacote do R, mas precisa de um programa, liberado apenas para teste
NotePad++: livre, mas não se comunica diretamente

ESS: *Emacs Speaks Statistics* (ESS), várias plataformas
(ótimo documento Musgo!)

TinnR: This is not a notepad, brasuca… (José Claudio Faria) UESC - Ilheus

Tinn R

http://sourceforge.net/projects/tinn-r

Vamor ver um exemplo:

Abrindo o simula.r no Tinn 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 ~ ALGORÍTMO DA COZINHA

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 de uma função (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

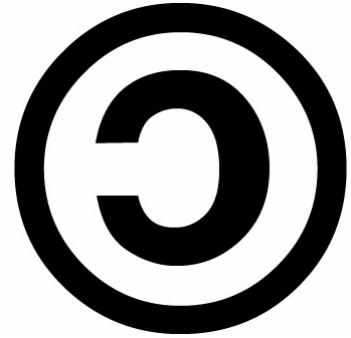
```
minha.funcao <- function( argumento1,
                  argumento2,. . .)
                {
                   comando 1
                   comando 2
                   comando 3 . . .
                   return()
                }
```

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!

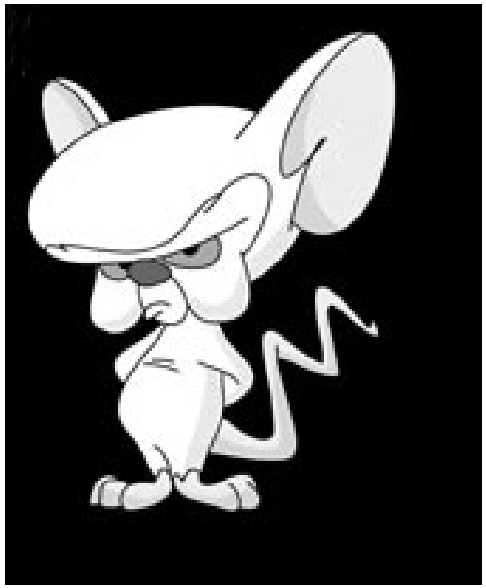
Código Livre.
Pode criar, modificar, copiar e distribuir.

General Public Licence (GNL,GNU)
Free Software Foundation

Primeira Lição!!

O que é mais comum a quem se aventuram a programar uma função???

Depois de fazer a primeira Função no R!



O mais comum!

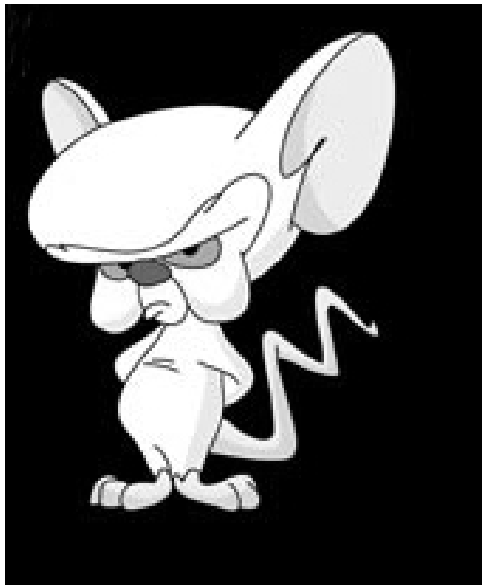
É uma versão piorada de uma função que já existe e vc. não sabia!

function ()

Funções Simples

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

Vamos ao R!



If(){}; else{}
Funções Simples

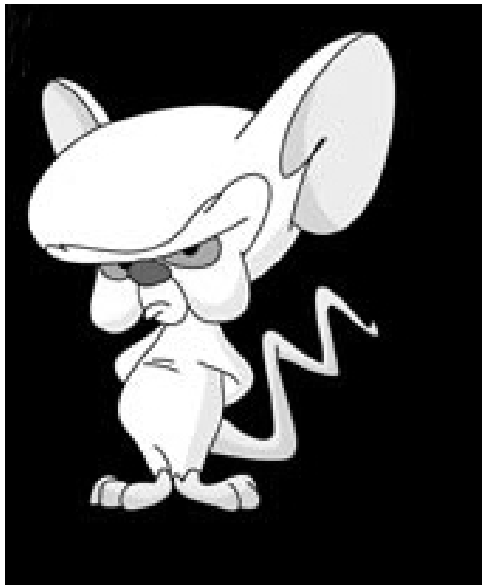
```
If(condição)
    {
    comandos se condição = TRUE
    }
else
    {
    comandos se condição = FALSE
    }
```

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

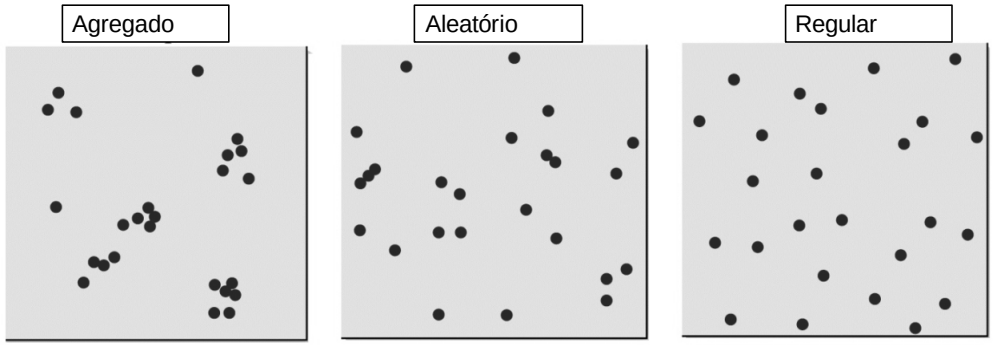
Biológia x Matemática

Uniforme: os valores observados estão próximos a média

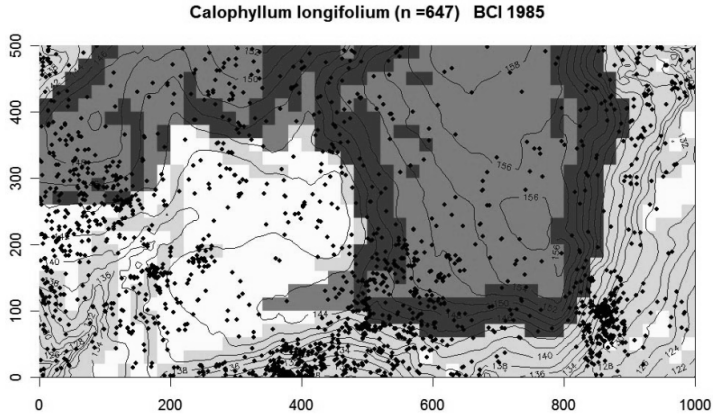
Aleatório: os valores observados estão dispersos aleatoriamente ao redor da média

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

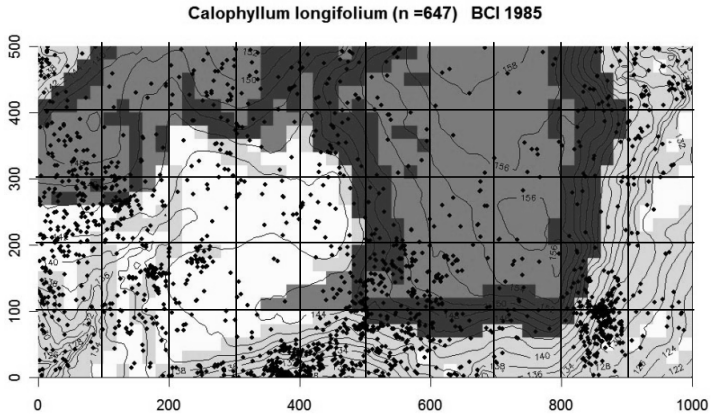
Padrões de distribuição



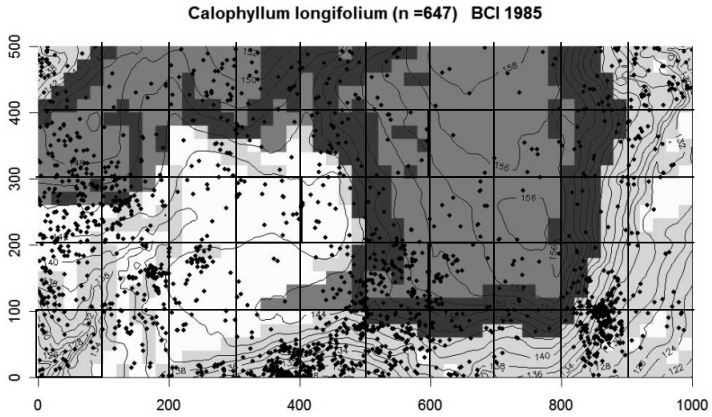
Distribuição Espacial



Distribuição Espacial

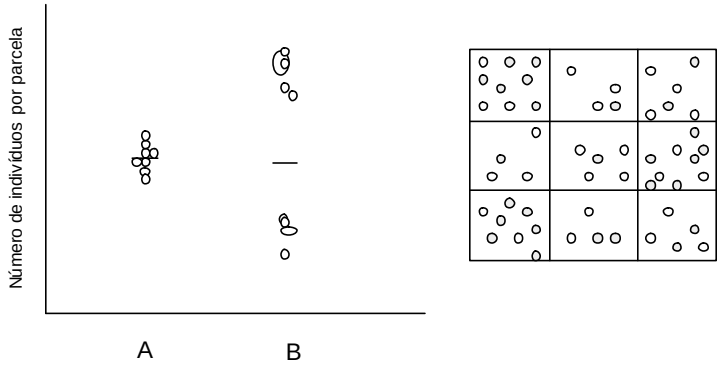


Prática – Sorteio



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)
    }
```

If(){}; else{}
Funções Simples

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

teste.ID ()
Funções Dispersão

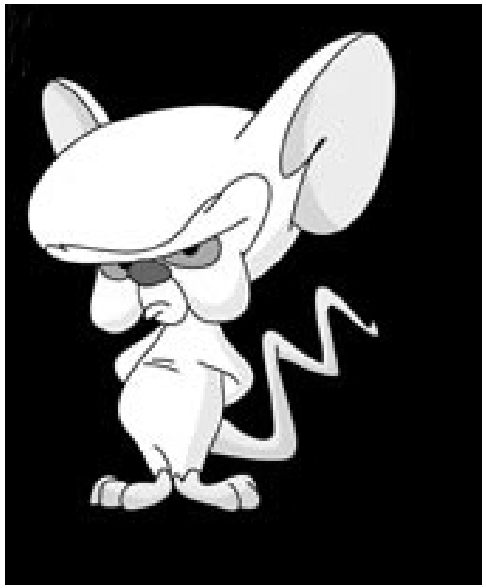
```
test.ID <- function(x)
 {
 dados=na.omit(x)
 med=media(dados)
 dev.quad=(dados-med)^2
 qui=sum(dev.quad)/med
 id=ID.curso(dados)
 critico.qui<- qchisq(c(0.025,0.975),df=(length(dados)-1))
        if(qui>=critico.qui[1] & qui<=critico.qui[2])
              {
              ("\n\t distribuição aleatória para
              alfa=0.05\n\t ID= “, id)
              }
         eles{}
...
```

teste.ID ()
Funções Dispersão

```
...
        if(qui < critico.qui[1])
             {
             cat("\n\t","distribuição regular, p<0.025 \n")
             }
        else{}

        if(qui>critico.qui[2])
             {
              cat("\t","distribuição aleatória, p>0.975 \n")
             }
      resulta=c(qui,critico.qui)
      names(resulta)<-c("qui-quadrado", "critico 0.025",
                                         "critico 0.975")
       return(resulta)
       }
```

Vamos ao R!



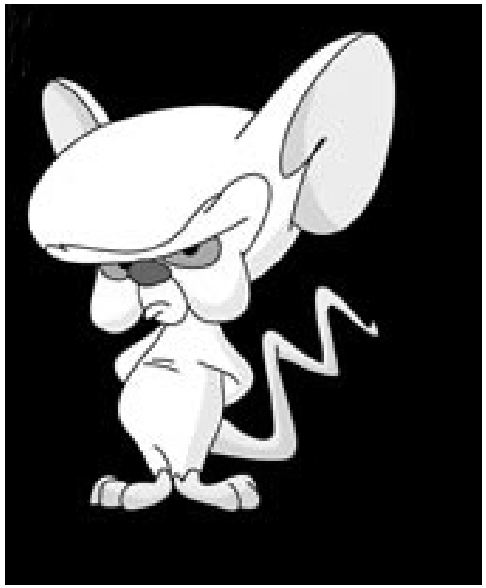
for(){}

Funções com ciclos

dados = matrix (spp , parcelas)

```
n.spp<-function(dados)
    {
    nplot=dim(dados)[2]
    resultados=rep(0,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!!!!)
Caso tenha conseguido “source” da função e não consegue entender o erro ao aplicar a função...
use:
debug(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 qual o formato de entrada

- Faça uma biblioteca de funções por aplicação (use o “salve 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

Depois de fazer a primeira Função no R!