Estatística Computacional

Patrícia de Siqueira Ramos

PPGEAB
UNIFAL-MG

28 de Março de 2018

Estruturas de controle de programação

- estrutura condicional:
  `if/else`
- estruturas repetitivas:
  - `while`
  - `for`
  - `repeat`

Condicional: if/else

- Estrutura:

```
if(condição){
  comandos
}else{
  comandos
}
```

Condicional: if/else

- Estrutura:

```
if(condição){
  comandos
}else{
  comandos
}
```

- Exemplo:

```
x = 1
if(x > 1){
  y = 5
}else{
  y = x
}
y
```

Condicional simples - exemplo em R

```
calor = F
temperatura = 35
if(temperatura > 30){
   calor = T
}
calor
```

Condicional composta - exemplo em R

```
media = 7
if(media >= 6){
   print('Você foi aprovado.')
}else{
   print('Você foi reprovado.')
}
```

Repetitiva: while

- Estrutura:

```
while(condição){
  comandos
}
```

Repetitiva: while

- Estrutura:

```
while(condição){
  comandos
}
```

- Exemplo:

```
i = 1
while(i <= 35){
  print(i)
  i = i + 1
}
```

Obs.: contadores devem ser atualizados.

Ex.: Potências de 2 usando Enquanto - exemplo em R

```
i = 0
while(i <= 10){
   print(2 ** i)
   i = i + 1
}
```

Repetitiva: repeat

- Estrutura:

```
repeat{
   comandos
   if(condição) break
}
```

Repetitiva: repeat

- Estrutura:

```
repeat{
  comandos
  if(condição) break
}
```

- Exemplo:

```
i = 1
repeat{
  print(i)
  i = i + 1
  if(i > 35) break
}
```

Obs.: contadores devem ser atualizados.

Repetitiva: for

- Estrutura:

```
for(i in seq){
  comandos
}
```

- Exemplo:

```
n = 10
x = matrix(0,n,1)
for(i in 1:n){
  aux = i * 2 + 3 - i
  x[i] = aux
}
x
```

Obs.: contadores não precisam ser atualizados.

Funções no R

- O conteúdo das funções prontas no R pode ser visto digitando o nome da função (sem os parênteses):
  `lm`
  `glm`
  `var`
- Mas esse recurso não está disponível desta forma para todas as funções como, por exemplo, em `min`, `max`, `rnorm` e `lines`
- Nesses casos, as funções não são escritas em linguagem R (em geral, estão escritas em C) e, para visualizar seu conteúdo, deve-se examinar os arquivos do código fonte do R.

Exemplo de função pronta do R

```
?sd  # sd(x, na.rm = FALSE)
dados = rnorm(100)  # gerar 100 valores da N(0,1)
# chamadas equivalentes da função sd
sd(dados)
sd(x = dados)
sd(x = dados, na.rm = F)
sd(na.rm = F, x = dados)
sd(dados, na.rm = F)
```

Ob.: NA significa *not available* (valores faltantes) e NAN significa *not a number*

Escrever funções próprias no R

- Sintaxe:

```
minha_funcao = function(argumento1, argumento2, ...){
  # algo interessante
}
```

- O valor de retorno de uma função é a última expressão que está nela
- Possuem argumentos que podem ter valores padrão
- Podem haver chamadas a outras funções dentro de uma função

Alguns exemplos de funções

```
f = function(a,b){
  return(a ** b)
}
f(2,3)  # uso

# argumentos não usados
f1 = function(a,b){
  return(a ** 2)
}
f1(2)  # uso

# mas
f2 = function(a,b){
  return(a ** b)
}
f2(45)  # uso
```

Exemplo - função em R que retorna a soma de n lançamentos de um dado

Como seria uma função em R que simule n lançamentos de um dado e retorne a soma das n faces?

Exemplo - função em R que retorna a soma de *n* lançamentos de um dado

Como seria uma função em R que simule *n* lançamentos de um dado e retorne a soma das *n* faces?

```
soma_dado = function(n) {
   k = sample(1:6, size=n, replace=TRUE)
   return(sum(k))
}
```

Exemplo - função em R que retorna a soma de *n* lançamentos de um dado

Como seria uma função em R que simule *n* lançamentos de um dado e retorne a soma das *n* faces?

```
soma_dado = function(n) {
   k = sample(1:6, size=n, replace=TRUE)
   return(sum(k))
}
```

a) Teste a função soma_dado.
b) Modifique a função para que ela também retorne quais foram os lançamentos.

Função para obter a moda de um conjunto de dados

```
moda = function(x){
    temp = table(x)
    return(names(temp)[temp == max(temp)])
}

# uso da função
y = c(1, 2, 2, 2, 3, 4, 5, 6, 6, 6)
moda(y)
```

Exemplo: teste t para a média

Queremos uma função para calcular a estatística do teste t para uma amostra:

$$H_0 : \mu = \mu_0 \qquad \times \qquad H_1 : \mu \neq \mu_0$$

$$\text{estatística do teste} : t_c = \frac{\bar{X} - \mu_0}{S/\sqrt{(n)}}$$

Exemplo: teste t para a média

Queremos uma função para calcular a estatística do teste t para uma amostra:

$$H_0 : \mu = \mu_0 \qquad \times \qquad H_1 : \mu \neq \mu_0$$

$$\text{estatística do teste} : t_c = \frac{\bar{X} - \mu_0}{S/\sqrt{(n)}}$$

Tal função deve retornar:

- o valor da estatística t_c
- a média amostral $\bar{X}$
- valor-p

Obs.: O R já possui a função `t.test` para isso.

Exemplo: teste *t* para a média em R

```
testet = function(x, mu0){
   n = length(x)
   S = sd(x)
   Xb = mean(x)
   tc = (Xb - mu0) / (S / sqrt(n))
   gl = n - 1
   valorp = 2 * (1 - pt(abs(tc), gl))
   list(tc=tc, media=Xb, valorp=valorp)
}
```

a) Use a função `testet` com a amostra [3.4, 5.6, 4, 6, 4.8, 9.1, 3.4, 4.5], testando se a média populacional pode ser considerada igual a 5.
b) Compare o resultado com a função `t.test` já implementada R

Modificação da função de teste t para a média

Já definir um valor padrão para μ_0 ($\mu_0 = 0$):

```
testet = function(x,mu0 = 0){
  n = length(x)
  S = sd(x)
  Xb = mean(x)
  t = (Xb - mu0) / (S / sqrt(n))
  gl = n - 1
  prob = 2 * (1 - pt(abs(t), gl))
  list(t=t, media=Xb, valorp=prob)
}
```

Teste a função com a mesma amostra do exemplo anterior, mas testando se $\mu_0 = 0$.

Exercícios

1. Criar uma função que receba como argumento um vetor com os dados e retorne o valor da média dos elementos (não use a função `mean` já pronta no `R`). Faça duas versões:
a) `media1` deve usar a estrutura de repetição `for` para ir somando cada número do vetor de dados.
b) `media2` pode usar a função `sum` pronta do `R` para somar os elementos do vetor.
c) Teste o uso de suas funções com:

```
d = rnorm(10, 5, 1)
media1(d)
media2(d)
mean(d)  # para verificar seus resultados
```

2. Modificar a função `testet` para retornar também os graus de liberdade.

Exercícios

3. Modificar a função `testet` para permitir testes unilaterais ou bilateral. Uma variável `tipo` deverá ter as opções: "maior", "menor" e "bilateral". O padrão deve ser o teste bilateral. A definição da função deve ser

`testet = function(x, mu0=0, tipo ="bilateral").`

Compare com os resultados da função `t.test` do R nos 3 casos. Teste o uso de sua função com:

```
w = rnorm(10, 3, 1)
tc1 = testet(w, mu0=5, tipo="maior")  # unilateral dir., mu0 = 5
tc1
t.test(w, mu=5, alternative="g") # função do R
tc2 = testet(w, mu0=5, tipo="menor") # unilateral esq., mu0 = 5
tc2
t.test(w, mu=5, alternative="l")
tc3 = testet(w, mu0=5, tipo="bilateral") # bilateral
tc3
t.test(w, mu=5)
```

Arquivos

- Muitas vezes, é necessário ler um arquivo de dados externo e carregá-lo no R
- O R é capaz de ler praticamente todos os tipos de arquivos (seja com o pacote básico, "base-r", ou com algum pacote adicional)
- As extensões de arquivos mais comuns são:
  .csv (cada célula é separada por vírgula ou ponto e vírgula)
  .txt (arquivo de texto)
  .dat (outro tipo de arquivo de texto)
  .RData (formato especial do R)
  .xls (planilha do Excel/Calc) - não usar!

Abrir arquivos .csv

```
getwd()  # visualizar a pasta ativa
setwd("~/Downloads")  # definir a pasta ativa
# ou ctrl + shift + H

#ler csv - atenção sep, dec e header (h)
emp = read.csv("emp_vga.csv",sep = ";",dec = ",",h = T)
emp
# visualizar tipos de dados
str(emp)
# transformar código do ibge em fator
emp$ibge7 = as.factor(emp$ibge7)

# salvar como .RData
save(emp,file="emp_vga.RData")
```

Abrir arquivos .RData

Vamos abrir o arquivo RData que criamos:

```
setwd("~/Downloads")  # definir a pasta ativa
# ou ctrl + shift + H
load("emp_vga.RData") # indicar o caminho do arquivo
# ou ctrl+shift+H
# ou aba Files do RStudio
emp
row.names(emp) = emp$ibge7
head(emp)
# coluna ibge7 ficou redundante - tirar
emp = emp[-1]
head(emp)
# se quisermos tirar também a coluna PBF
emp = emp[-2]
# se quisermos criar outro data frame
emp2 = emp
head(emp2)
# salvar como .RData
save(emp2,file="emp2_vga.RData")
```

Abrir arquivo atlas.csv

```
atlas = read.csv("~/Downloads/atlas.csv",h = T)
atlas
# mostrar apenas 20 variáveis
head(atlas[,1:20])
# criar um dataframe com as 20 variáveis
dados = atlas[,1:20]
dim(dados)
# apenas MG
mg = subset(dados,uf == 31)
dim(mg)
head(mg)
# salvar como rdata
# mudar a pasta se necessário
setwd("~/Downloads/")
save(mg, file = "mg.Rdata")

# carregar conjunto de dados salvo
# ou definir a pasta primeiro com setwd()
load("mg.Rdata")
# ou ir em Files do RStudio
```

Abrir outros tipos de arquivos

- Para trabalhar com arquivos `.txt` ou `.dat`, basta usar os comandos:

```
# ler
dados = read.table("nomearquivo.txt",h = T)
dados
# salvar
write.table(dados2,file = "nomearquivo2.txt")
# para arquivos .dat o procedimento é o mesmo
```

- Há alguns pacotes que podem ajudar na tarefa de abrir diferentes tipos de arquivos:
  `readr`: read_csv e variantes
  `readxl`: read_excel
  `haven`: sas, spss, stata

Exercícios com arquivos

4. No site do IMRS (http://imrs.fjp.mg.gov.br/) criar outro csv com variáveis interessantes, abrí-lo no R, salvar como .RData e abrir o RData criado.