MAC0113-aula23

Hirata

26/06/2021

Aula 23

Sejam bem vindas, sejam bem vindos!

O link do slido de hoje é:

(turma 21): https://app.sli.do/event/mxgnxdhe

(turma 22): https://app.sli.do/event/vuxs58hi

Objetivos de hoje

- O que é uma string: definição, fatiamento e comprimento
- imutabilidade de strings
- Codificação de caracteres
- Leitura de arquivo de texto linha a linha
- R Markdown

Strings

- Uma string é uma sequência de caracteres
- Para criar uma string, usamos as aspas simples, ou duplas

```
texto <- "Este é um texto curto."

texto2 <- 'Este é um texto curto.'
```

- Mas não pode-se misturar as aspas simples com duplas

Strings

- Strings podem ser concatenadas com o comando paste

```
texto1 <- "Olá,"

texto2 <- "meu"

texto3 <- "nome é"

texto4 <- "Roberto Hirata Jr."

frase <- paste(texto1,texto2,texto3,texto4)

print(frase)
```

```
## [1] "Olá, meu nome é Roberto Hirata Jr."
```

Strings

- O comando paste concatena com o separador " ", mas ele pode ser mudado:

```
texto1 <- "Olá,"

texto2 <- "meu"

texto3 <- "nome é"

texto4 <- "Roberto Hirata Jr."

frase <- paste(texto1,texto2,texto3,texto4,sep="***")

print(frase)
```

```
## [1] "Olá,***meu***nome é***Roberto Hirata Jr."
```

Strings

- O comando paste concatena com o separador " ", mas ele pode ser mudado:

```
texto1 <- "Ba"

texto2 <- "na"

texto3 <- "na"

palavra <- paste(texto1,texto2,texto3,sep="")

print(palavra)
```

```
## [1] "Banana"
```

Tamanho de uma string

- O número de caracteres de uma string pode ser conseguido com a função nchar

```
texto1 <- "Hoje está frio! Está"

temperatura <-"17\u00BAC!"

frase <- paste(texto1,temperatura,sep=" ")

print(frase)
```

```
## [1] "Hoje está frio! Está 17ºC!"
```

```
cat("Essa frase tem ",nchar(frase)," caracteres.")
```

```
## Essa frase tem  26  caracteres.
```

Tamanho de outros tipos de dados

- A função nchar pode ser usada para contar o número de dígitos de um número

```
preco <- 123987

print(typeof(preco))
```

```
## [1] "double"
```

```
cat("O número ",preco," é do tipo ",typeof(preco),
    " e tem ",nchar(preco)," dígitos.")
```

```
## O número  123987  é do tipo  double  e tem  6  dígitos.
```

Codificação de caracteres

- Como vimos em uma aula anterior, dentro da memória do computador são armazenados apenas números
- Para representar caracteres, precisamos codificá-los como números.
- Vimos duas formas de codificação ASCII e UTF8
- Nesta URL podemos ver as codificações do UTF8

https://www.utf8-chartable.de/unicode-utf8-table.pl?number=1024

```
textoutf8 <- "\u00BC \u00BD \u00BE."

print(textoutf8)
```

```
## [1] "¼ ½ ¾."
```

Codificação de caracteres e comparação de strings

- Duas string são iguais se elas são iguais caractere a caractere
- Dois caracteres são iguais se eles têm a mesma codificação

```
texto = "banana"
texto2 = "laranja"
print(texto==texto2)
```

```
## [1] FALSE
```

```
texto3 = "Banana"
print(texto==texto2)
```

```
## [1] FALSE
```

```
texto4 = "banana"
print(texto==texto4)
```

```
## [1] TRUE
```

Codificação de caracteres e comparação de strings

- Para facilitar a comparação de strings, uma estratégia é transformar as strings previamente para caixa alta (maiúsculas)
- A função toupper transforma uma string para maiúsculas e a função tulower para minúsculas

```
texto = "BanAna"
print(toupper(texto))
```

```
## [1] "BANANA"
```

```
print(tolower(texto))
```

```
## [1] "banana"
```

```
texto3 = "Banana"
print(toupper(texto)==toupper(texto3))
```

```
## [1] TRUE
```

Substrings

- Algumas vezes podemos querer pegar apenas uma parte da string
- Isso pode ser feito com a função substring

```
texto = "MAC0113"
departamento = substring(texto,1,3)
codigoNum = substring(texto,4,nchar(texto))
cat("O departamento é:",departamento,
    " e o código é:",codigoNum)
```

```
## O departamento é: MAC  e o código é: 0113
```

Substrings

- A função substring também serve para substituirmos parte de uma string

```
texto = "MAC0113"
substring(texto,4,nchar(texto)) <- "0115"
cat("A nova string é:",texto)
```

```
## A nova string é: MAC0115
```

Procurando uma string dentro de uma string

- Veja o help da função grep

```
seqProt = "GHLHGKHHCGHGCAGALLCAAAMMKHMGLKHGAGAGKGCA
CKMALLKALMKCGLMLGCLGHMHGCGAAKHMCLCMLAGKAALGLCMAHHMA
GLLMHMMLK"
seqex <- "ACHKLMG"
seqex2 = "HGKHHCGH"
print(grepl(seqex,seqProt))
```

```
## [1] FALSE
```

```
print(grepl(seqex2,seqProt))
```

```
## [1] TRUE
```

Quebrando uma frase em palavras

- A função strsplit pode ser usada para separar as palavras de uma frase.

```
ajuda = "Texto curto para exemplificar o strplit"
palavras = unlist(strsplit(ajuda," "))
for (palavra in palavras) {
  cat(palavra," ")
}
```

```
## Texto  curto  para  exemplificar  o  strplit
```

Lendo um arquivo texto linha a linha

- A função readLines lê um arquivo linha a linha

```
texto = readLines("emailDesesperado.txt")
print(length(texto))
```

```
## [1] 18
```

Lendo um arquivo texto linha a linha

```
conta = 1
for (linha in texto)
  if (nchar(linha)!=0) {
    if (conta%%9!=0) {
      cat(nchar(linha)," ")
    } else {
      cat(nchar(linha),"\n")
    }
    conta = conta + 1
  }
```

```
## 234  553  495  88  163  281  284  211  130
## 127  69  558
```

Lendo um arquivo texto palavra por palavra

- A função scan lê um arquivo palavra por palavra

```
texto2 = scan("emailDesesperado.txt",quote=NULL,what="x")
print(length(texto2))
```

```
## [1] 514
```

Lendo um arquivo texto palavra por palavra

```
conta = 1
for (palavra in texto2)
  if (nchar(palavra)>8) {
    if (conta%%15!=0) {
      cat(nchar(palavra)," ")
    } else {
      cat(nchar(palavra),"\n")
    }
    conta = conta + 1
  }
```

```
## 13  11  12  11  10  10  9  11  16  11  11  11  10  9  9
## 10  13  13  11  10  9  12  9  9  9  13  10  9  10  11
## 9  9  12  9  10  9  9  10  10  12  10  10  9  11  9
## 9  12  11  10  18  11  11  12  10  9  9  11  9  16  10
## 9  12  11  9  10  9  12  9  12  11  10  9  9  10  10
## 18  9  15  10  11  11  10  9  11  13  10  9  10  9  10
## 11  11  12
```

R Markdown

- Esta apresentação foi feita inteiramente em R Markdown.
- Semelhante a um hipertexto, o R Markdown tem marcações para fazer apresentações bonitas e juntar código em R.
- A saída pode ser um arquivo HTML, ou PDF, ou MS Word.
- Se você quiser saber mais sobre este assunto, olhe:

– http://rmarkdown.rstudio.com

– https://bookdown.org/yihui/rmarkdown-cookbook/

R Markdown no RStudio

- Se você usar o RStudio, você pode criar um arquivo R Markdown
- Uma vez criado, você pode apertar o botão **Knit** e o documento será gerado

Vantagens de usar o R Markdown

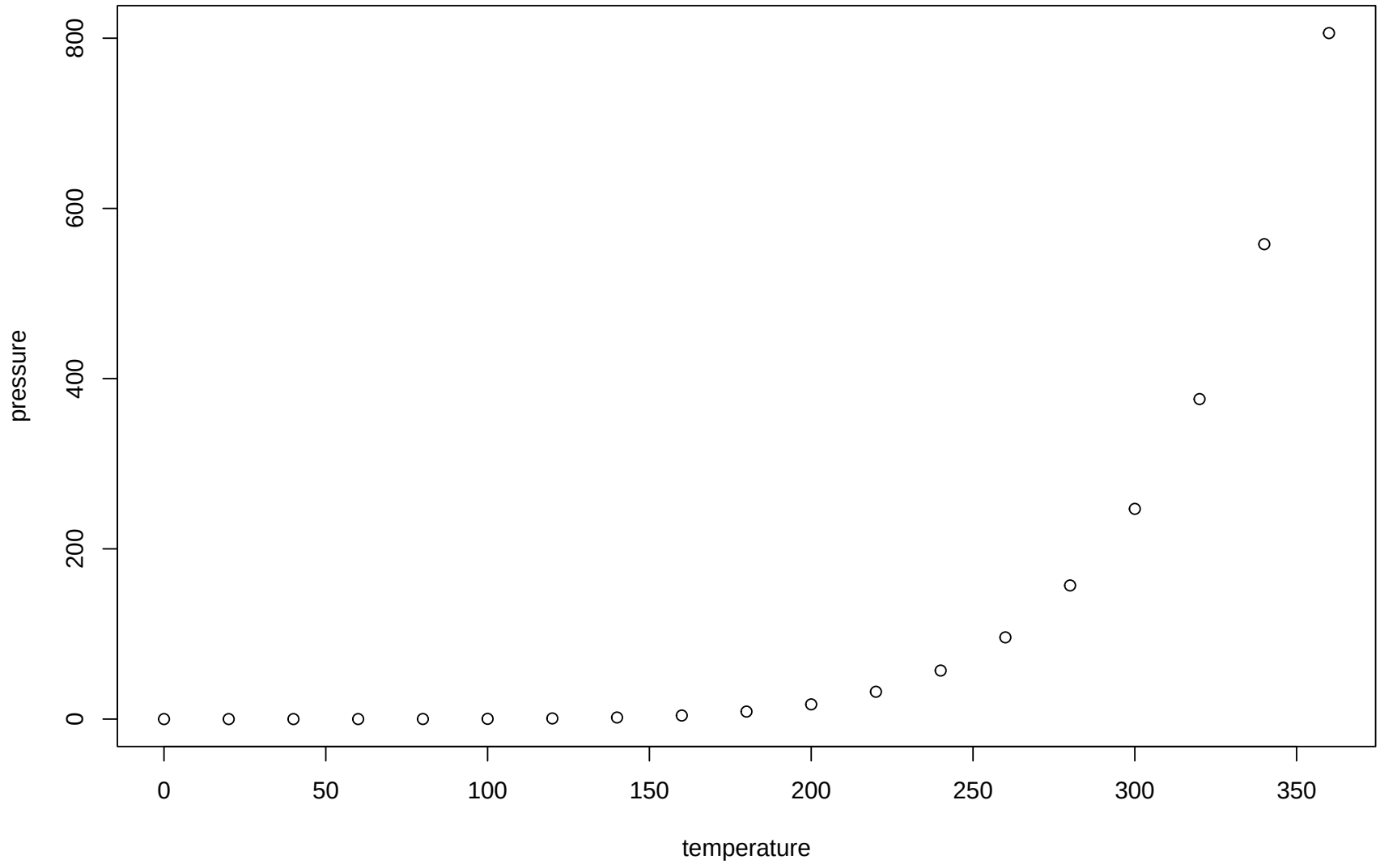
- Você pode incluir tabelas apenas carregando-as

```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

Vantagens de usar o R Markdown

- Você pode fazer gráficos e juntar nas suas apresentações, ou seus relatórios



Obrigado!