Prova de seleão - R in Rio

Julia Ferreira

Apresentação

Os códigos e resultados foram elaborados por Julia Hellen Ferreira. Os pacotes necessários para resolução dos problemas foram o "Tidyverse", para manipulaçao da base de dados, e o "Rio" para importar a base de dados. Alguns resultados foram encurtados para não ocupar muito espaço. No e-mail segue o script onde todos os resultados poderão ser vistos de forma completa.

Questão 1

Nesta questão, a função deveria retornar os k maiores números de um determinado vetor porém encontrei certa dificuldade para dar a possibilidade de escolha. Então, fiz uma função que irá receber um vetor númerico e retornar o maior número do vetor.

```
v <- NULL
max <- function(v){
  maior <- v[1]
  for(i in 1:length(v)){
    if(maior < v[i]){
      maior <- v[i]
    }
  }
  return(maior)
}
max(c(1,2,600,5,10,500)) # Exemplo
```

```
## [1] 600
```

Questão 2

Para a questão 2, foi necessário gerar um data.frame com os 100 priemiros números inteiros e para cada uma deles usei a função rnorm() com os seguintes parâmetros: n = 100 e a média deverá ser inserida. Mas deixei media = 10 com default da função. A função irá receber a média e retornar um data.frame.

```
Num_Int = seq(1:100)
df <- as.data.frame(Num_Int)

normal <- function(media = 10){
  for (i in 1:100){
    for (j in 2:101) {
    lista <- rnorm(100, media)
    df[i,j] <- lista[i]
  }
  }
  return(df)
}
```

```r
resultado <- normal(10) # Exemplo
head(resultado[,10, 10])
```

```
## [1]  9.837913 12.559515  9.413243 10.137875 10.625464  7.601543
```

Questão 3

Na questão 3, eu usei funções do "Tidyverse" para realizar a tarefa de montar um novo tibble. A função gather() foi usada para transformar as variaveis em observações. E a função spread() foi para fazer o oposto, tranformar observações em colunas independentes.

```r
library(tidyverse)
library(dplyr)
library(rio)

data <- import("https://www.rug.nl/ggdc/docs/10sd_jan15_2014.xlsx",
               sheet = 2,
               setclass = "tibble") %>%
    filter(Variable == "VA_Q05" | Variable == "EMP") %>%
    select(-"Region code", -"Summation of sector GDP") %>%
    gather(c(Agriculture, Mining, Manufacturing, Utilities, Utilities, Construction,
             `Trade, restaurants and hotels`, `Transport, storage and communication`,
             `Finance, insurance, real estate and business services`,
             `Government services`, `Community, social and personal services`),
           key = "Sector", value = "Valor") %>%
  spread(key = Variable, value = Valor)

data # resultado final
```

```
## # A tibble: 25,350 x 6
##    Country Region       Year Sector                                    EMP VA_Q05
##    <chr>   <chr>       <dbl> <chr>                                   <dbl>  <dbl>
##  1 ARG     Latin Amer~  1950 Agriculture                           1800.  16179.
##  2 ARG     Latin Amer~  1950 Community, social and personal servi~  411.   7018.
##  3 ARG     Latin Amer~  1950 Construction                           314.   9171.
##  4 ARG     Latin Amer~  1950 Finance, insurance, real estate and ~  204.   6360.
##  5 ARG     Latin Amer~  1950 Government services                    825.  23893.
##  6 ARG     Latin Amer~  1950 Manufacturing                         1603.  40416.
##  7 ARG     Latin Amer~  1950 Mining                                  32.7  1993.
##  8 ARG     Latin Amer~  1950 Trade, restaurants and hotels          890.  24403.
##  9 ARG     Latin Amer~  1950 Transport, storage and communication   425.   7550.
## 10 ARG     Latin Amer~  1950 Utilities                               39.3   574.
## # ... with 25,340 more rows
```

Questão 4

Minha ideia em um primeiro momento para resolução desta questão foi transformar o vetor em um data.frame dentro da função. Assim eu consegui trabalhar de forma mais livre. A função irá receber um vetor e retornar a média (Vetor númerico), um data.frame com o character que mais se repetiu ou número de vezes (Vetor com character) ou a média de entradas únicas que mais se repetiu (Vetor com character).

```r
library(tidyverse)

teste <- function(vetor){
  if (is.numeric(vetor)){
    resposta = mean(vetor)
```

```
  }
  if (is.character(vetor)){
    for (i in 1:length(vetor)){
      vetor1 <- as.data.frame(vetor)
      contagem <- vetor1 %>%
        group_by(vetor) %>%
        summarise(n = n())
     maior <- as.numeric(max(contagem$n))
     contagem <- contagem %>%
       filter(contagem$n == maior)
     if(contagem$n > length(vetor)/3){
       resposta <- maior/length(vetor)
     }
     else{
       resposta <- contagem %>%
       filter(contagem$n == maior)
     }
    }
  }
  return(resposta)
}

teste(c(1,2,3)) # Exemplo
```

```
## [1] 2
```

```
teste(c("a","a","b")) # Exemplo
```

```
## [1] 0.6666667
```

Questão 5

Por fim, fiz uma função da sequência de Fibonacci que irá receber um número de parada e retornar a sequência até o número inserido.

```
fib <- function(n){
  f <- c(1,1)
  for(i in 3: n){
    f[i] <- f[i - 1] + f[i-2]
  }
  return(f)
}

fib(12)
```

```
##  [1]   1   1   2   3   5   8  13  21  34  55  89 144
```

Fiz todas as questões obrigatórias da forma que pensei ser mais fácil e com os conheciemntos que tenho hoje. Talvez eu veja que não seja o melhor caminho e com isso vou aprimorando meus códigos a cada dia.