Lista de Exercícios 2

Thaís Paiva

14/04/2023

Funções no R

Exercício 1

```
seqrep = function(n){
  return( rep(1:n, times=1:n) )
}

length(seqrep(50))
```

```
## [1] 1275
```

Exercício 2

```
maior.xbarra = function(x){
  return( x[x>mean(x)] )
}
```

Exercício 3

```
maior.xbarra(seqrep(10))
```

```
##  [1]  8  8  8  8  8  8  8  8  9  9  9  9  9  9  9  9 10 10 10 10 10 10 10 10
## [26] 10 10
```

Exercício 4

```
i = 10:20
sum( i^2 + 4/i )
```

```
## [1] 2588.075
```

Exercício 5

```
s = NULL
for(i in 1:10){
  for(j in 1:10){
    s = sum(s, i^2/(5 + i*j) )
#    cat("i: ",i,"j: ",j," soma: ",s,"\n")
  }
}
s
```

```
## [1] 117.4334
```

Exercício 6

```
a = c(-0.2, 0.2, 0.49, 0.5, 0.51, .99, 1.2)
trunc(a)
```

```
## [1] 0 0 0 0 0 0 1
```

```
floor(a)
```

```
## [1] -1  0  0  0  0  0  1
```

```
ceiling(a)
```

```
## [1] 0 1 1 1 1 1 2
```

```
round(a)
```

```
## [1] 0 0 0 0 1 1 1
```

A função `trunc(a)` pega a parte inteira dos números de `a`. A função `floor(a)` retorna o maior número inteiro menor do que os números de `a`, já a função `ceiling(a)` retorna o menor número inteiro maior do que os números de `a`. Por último, a função `round(a)` arredonda os números de `a` com o número de digitos especificados, no caso, zero.

Exercício 7

```
set.seed(1)
x = runif(20,-1,1)
ifelse(x>0, log(x), NA)
```

```
## Warning in log(x): NaNs produzidos
```

```
##  [1]          NA          NA -1.92615940 -0.20283176          NA -0.22717746
##  [7] -0.11726382 -1.13446047 -1.35391202          NA          NA          NA
## [13] -0.98337731          NA -0.61677365          NA -0.83186453 -0.01632026
## [19]          NA -0.58898459
```

Exercício 8

```
paste0("Seg",1:100)
```

```
##   [1] "Seg1"   "Seg2"   "Seg3"   "Seg4"   "Seg5"   "Seg6"   "Seg7"   "Seg8"
##   [9] "Seg9"   "Seg10"  "Seg11"  "Seg12"  "Seg13"  "Seg14"  "Seg15"  "Seg16"
##  [17] "Seg17"  "Seg18"  "Seg19"  "Seg20"  "Seg21"  "Seg22"  "Seg23"  "Seg24"
##  [25] "Seg25"  "Seg26"  "Seg27"  "Seg28"  "Seg29"  "Seg30"  "Seg31"  "Seg32"
##  [33] "Seg33"  "Seg34"  "Seg35"  "Seg36"  "Seg37"  "Seg38"  "Seg39"  "Seg40"
##  [41] "Seg41"  "Seg42"  "Seg43"  "Seg44"  "Seg45"  "Seg46"  "Seg47"  "Seg48"
##  [49] "Seg49"  "Seg50"  "Seg51"  "Seg52"  "Seg53"  "Seg54"  "Seg55"  "Seg56"
##  [57] "Seg57"  "Seg58"  "Seg59"  "Seg60"  "Seg61"  "Seg62"  "Seg63"  "Seg64"
##  [65] "Seg65"  "Seg66"  "Seg67"  "Seg68"  "Seg69"  "Seg70"  "Seg71"  "Seg72"
##  [73] "Seg73"  "Seg74"  "Seg75"  "Seg76"  "Seg77"  "Seg78"  "Seg79"  "Seg80"
##  [81] "Seg81"  "Seg82"  "Seg83"  "Seg84"  "Seg85"  "Seg86"  "Seg87"  "Seg88"
##  [89] "Seg89"  "Seg90"  "Seg91"  "Seg92"  "Seg93"  "Seg94"  "Seg95"  "Seg96"
##  [97] "Seg97"  "Seg98"  "Seg99"  "Seg100"
```

Base de Dados

Exercício 9

```
require(hmmm)
data("accident")
```

```
nrow(accident)
```

```
## [1] 72
```

```
sum(accident$Freq)
```

```
## [1] 1052
```

O banco de dados possui 72 linhas. Como as covariáveis são todas categóricas, cada linha representa uma categoria de tipo de acidente (`uncertain`, `avoidable`, `not-avoidable`), tempo em dias que o empregado ficou afastado (`0-7`, `7-21`, `21-60`, `>60`), faixa etária do empregado (`<=25`, `26-45`, `>45`), e o período do dia em que o acidente ocorreu (`morning`, `afternoon`). Para cada combinação entre as covariáveis, foi registrado o número de acidentes com aquelas características na coluna `Freq`. Assim, o número total de acidentes é 1052.

Exercício 10

```
attach(accident)
mean(Freq[Type=="uncertain"])
```

```
## [1] 24.08333
```

O número médio de acidentes do tipo `uncertain` por categoria das demais covariáveis é de 24.0833333.

A combinação de variáveis com o maior número de acidentes do tipo `avoidable` é dada por:

```
accident[ which( Freq == max(Freq[Type=="avoidable"]) ), ]
```

```
##         Type    Time      Age    Hour Freq
## 14 avoidable 0 |-- 7 26 -- 45 morning   51
```

ou seja, o maior número de acidentes evitáveis foi registrado para acidentes que ocorreram na manhã, com empregados com idade entre 26 e 45 anos, e que ficaram afastados por menos de uma semana.