Lista nº 3 – Funções

Instruções para entrega da lista:

a) A resolução da lista deve estar apresentada em documento extensão `.pdf` ou `.html`, gerada em **R Markdown**.
b) O arquivo com o relatório de respostas deverá ser denominado `065-243_L03-SEUNOME`.
c) Não esqueça de **se identificar no preâmbulo do arquivo**, além de rotular corretamente as questões cujos comandos e resultados você apresentará. IMPORTANTE: **use apenas funções disponíveis no R Base**.
d) Apresente todos os comandos (todos os comandos que funcionaram!) que utilizou para obter os resultados solicitados. Se alguma questão não deu certo, apresente assim mesmo seus comandos, não se esquecendo de indicar usando dois jogos da velha (`##`) em cada um de suas linhas.
e) **Preserve a ordem** das questões e responda brevemente suas justificativas e comentários.
f) Não hesite em procurar o **Fórum de Dúvidas** do Moodle, caso tenha alguma dúvida com relação à solução da presente lista de exercícios. Caso não resolva, acione o professor. Acostume-se a interagir para obter sugestões de solução das dúvidas.

1. O *data frame* `fumig {DAAGxtras}` apresenta dados de uma série de experimentos em que o produto foi exposto a um inseticida durante um período de tempo de 2 horas. As concentrações do inseticida foram medidas nos tempos 5, 10, 30, 60, 90 e 120 minutos. O código fornecido abaixo calcula um produto concentração-tempo (`c-t`) que mede a exposição ao inseticida, levando à medida `ctsum`. Verifique o código das três alternativas de função dadas abaixo e o *data frame* `fumig` que é o *default* para o argumento `df`.
    a. Aplique todas as três funções ao conjunto de dados em questão (`fumig`) e verifique se elas apresentam o mesmo resultado.
    b. Documente o código da função `calcCT1`, explicando o que cada linha faz.
    c. Faça uma verificação visual e comente se as medições de concentração do inseticida são mais variáveis em alguns momentos do que em outros?
    d. Qual função é mais rápida? [Verifique a possível diferença, repetindo a execução da função 1000 ou mais vezes.].

Código de 3 funções que efetuam cálculos

```
> ## Function "calcCT1"
> "calcCT1" <-
+ function(df=fumig, times=c(5,10,30,60,90,120),
+ ctcols=3:8){
+     multiplier <- c(7.5,12.5,25,30,30,15)
+     m <- dim(df)[1]
+     ctsum <- numeric(m)
+ for(i in 1:m){
+     y <- unlist(df[i, ctcols])
+     ctsum[i] <- sum(multiplier*y)/60
+ }
```

```
+ df <- cbind(ctsum=ctsum, df[,-ctcols])
+ df
+ }
> ##
> ## Function "calcCT2"
> "calcCT2" <-
+ function(df=fumig, times=c(5,10,30,60,90,120),
ctcols=3:8){
+ multiplier <- c(7.5,12.5,25,30,30,15)
+ mat <- as.matrix(df[, ctcols])
+ ctsum <- mat%*%multiplier/60
+ cbind(ctsum=ctsum, df[,-ctcols])
+ }
> ##
> ## Function "calcCT3"
> "calcCT3" <-
+ function(df=fumig, times=c(5,10,30,60,90,120),
ctcols=3:8){
+ multiplier <- c(7.5,12.5,25,30,30,15)
+ mat <- as.matrix(df[, ctcols])
+ ctsum <- apply(mat, 1,
+ function(x)sum(x*multiplier))/60
+ cbind(ctsum=ctsum, df[,-ctcols])
+ }
```

2. A expressão da congruência de Zeller é dada por:

$$f = [2,6m - 0,2] + k + y + [y/4] + [c/4] - 2c) \mod 7$$

em que:

`[x]`: parte inteira de `x`. (ex.: `[7,5] = 7`).

A congruência de Zeller retorna o dia da semana (f) dados:

`k`: dia do mês,

`y`: o ano no século

`c`: os primeiros 2 dígitos do ano (o número do século)

`m`: o número do mês (onde janeiro é o mês 11 do ano anterior, fevereiro é o mês 12 do ano anterior, março é o mês 1, etc.)

Por exemplo, a data 21/07/1963 tem `m = 5`, `k = 21`, `c = 19`, `y = 63`; enquanto a data 21/02/1962 tem `m = 12`, `k = 21`, `c = 19` e `y = 62`.

   a. Escreva uma função `dia.semana(dia, mes, ano)` que retorne o dia da semana quando receber as entradas numéricas do dia, mês e ano. Observe que `f = 1` denota domingo; `f = 2`, segunda-feira etc.
   b. Sua função funciona se os argumentos de entrada: dia, mês e ano forem vetores (de mesmo tamanho e com entradas válidas)?

3. Funções diversas:
   a. Crie uma função que, dado um vetor numérico **x**, retorna os dígitos (de 0 a 9) que não estão em **x**. Ex.: Se **x** `= (0, 2, 4, 8)` a função deve retornar `(1, 3, 5, 6, 7, 9)`.
   b. Crie uma função em que dadas duas *strings* (com apenas uma palavra cada), seja verificado se uma *string* é um anagrama da outra.

c. Crie uma função em que, dada uma palavra, seja retornado a posição das letras dessa palavra no alfabeto. Por exemplo, se a palavra for `"abc"`, a função retornará `1 2 3`.

4. *CPF*. Por meio da lógica matemática do funcionamento do Cadastro de Pessoas Físicas (CPF), utilizado pela Secretaria Especial da Receita Federal do Brasil é possível identificar a região do CPF. Colocando de outra maneira, dado um número do CPF pode-se saber a Região Fiscal na qual ele foi emitido. O número de inscrição no Cadastro de Pessoas Físicas tem onze dígitos. Os oitos primeiros dígitos decimais formam um número-base definido pela receita Federal no momento da inscrição. O nono dígito define a Região Fiscal responsável pela inscrição. O penúltimo, é o dígito verificador dos nove primeiros. O último, é o dígito verificador dos nove anteriores a ele. Consulte o post "A Matemática nos Documentos: A Matemática dos CPF´s" e construa uma função que retorne a Região Fiscal de emissão de qualquer CPF e confira os dígitos verificadores (dois últimos algarismos). De maneira geral, os Dígitos Verificadores (DV) são dígitos incorporados a números para possibilitar a detecção de erros de digitação. Sua função deve aceitar como entrada apenas as *strings* no formato **ABC.DEF.GHI-JK**, sendo que as letras de A a K são algarismos decimais. Valide sua função usando o número de seu CPF (não precisa apresentar seu CPF!) e determine a região Fiscal de emissão e os dígitos verificadores do CPF 040.148.078-XY. O post citado acima é encontrado na url abaixo:

   — http://clubes.obmep.org.br/blog/a-matematica-nos-documentos-a-matematica-dos-cpfs/.