Questão B010 – A proposta é transformar a solução dos exercícios apresentados em aula do paradigma imperativo (com uso de loop) para o funcional ou matricial/vetorizado.

```
#-------------------------------------------------------------------------------------------#
#*******************************************************************************************#
#---------------------------------- Aula Repeticao1.R --------------------------------------#
#*******************************************************************************************#
#-------------------------------------------------------------------------------------------#
# Imprimir Bom dia na tela
N <- 5 # quantidade de vezes
for(i in 1:N) {
   cat("Bom dia R!", i, "\n")
}
```

```
#-------------------------------------------------------------------------------------------#
#*******************************************************************************************#
#---------------------------------- Aula Repeticao2.R --------------------------------------#
#*******************************************************************************************#
#-------------------------------------------------------------------------------------------#
# Algoritmo de soma em um vetor
vetor <- c(1, 2, 34, 12, 2)
N <- length(vetor)
soma <- 0
for (i in 1:N) {
   soma <- soma + vetor[i]
}
```

```r
#-------------------------------------------------------------------------------------------#
#*******************************************************************************************#
#---------------------------------- Aula Repeticao3.R --------------------------------------#
#*******************************************************************************************#
#-------------------------------------------------------------------------------------------#
# Obter o quadrado de elementos de um vetor
vetor <- c(1, 2, 34, 12, 2)
# Apenas imprimir na tela
for (i in 1:length(vetor)) {
   print(vetor[i]^2)
}

# Armazenar o resultado em um vetor
# Estratégia Modificar vetor
vetor <- c(1, 2, 34, 12, 2)
vet_quad <- vector("numeric", length(vetor))
for (i in 1:length(vetor)) {
   quad <- vetor[i]^2
   vet_quad[i] <- quad
}

# Divisíveis 4
vetor <- c(20, 2, 34, 12, 2)
div4_log <- vector("logical", length(vetor))
for (i in 1:length(vetor)) {
   if (vetor[i] %% 4 == 0) {
      div4_log[i] <- TRUE # verdadeiro
   }
}
div4_log
div4 <- vetor[div4_log]
```

```r
#-------------------------------------------------------------------------------------------#
#*******************************************************************************************#
#---------------------------------- Aula Estruturas_dados1.R -------------------------------#
#*******************************************************************************************#
#-------------------------------------------------------------------------------------------#
# Ajuste de modelos
# importar dados
library(readxl)
dados <- read_excel("G:/Meu Drive/Drive/Aulas/Algoritmos/Aulas2020_1remoto/dados/dados.xlsx")
# identificar quantidade de ajustes (repetições)
uni_gen <- unique(dados$genotipo)
N <- length(uni_gen)
# estrutura tibble para receber parametros do modelo
require(tibble)
coefs <- tibble(genotipo = vector("character", N),
                b0 = vector("numeric", N),
                b1 = vector("numeric", N))
# como acessar os elementos de interesse
# Processo de repetição (loop for)
for (i in 1:N) {
   dados_i <- subset(dados, genotipo == uni_gen[i])
   lm_i <- lm(volume ~ idade, dados_i)
   coefs[i, "genotipo"] <- uni_gen[i]
   coefs[i, "b0"] <- lm_i$coefficients[[1]]
   coefs[i, "b1"] <- lm_i$coefficients[[2]]
}
```

```r
#-------------------------------------------------------------------------------------------#
#*******************************************************************************************#
#---------------------------------- Aula Estruturas_dados2.R -------------------------------#
#*******************************************************************************************#
#-------------------------------------------------------------------------------------------#
# Ajuste de modelo não linear e plotar ajuste
library(minpack.lm)
library(readxl)
dados <- read_excel("G:/Meu Drive/Drive/Aulas/Algoritmos/Aulas2020_1remoto/dados/dados.xlsx")
prod <- c("G20", "G22", "G4", "G3", "G2", "G24", "G7", "G18")
inter <- c("G5", "G12", "G9", "G21", "G14", "G11", "G8", "G6")
resis <- c("G1", "G16", "G23", "G15", "G13", "G10", "G19", "G17")
dados$info_gen <- ""
dados[dados$genotipo %in% prod, "info_gen"] <- "produtivos"
dados[dados$genotipo %in% inter, "info_gen"] <- "intermediários"
dados[dados$genotipo %in% resis, "info_gen"] <- "resistentes"
# 1. Identificar e quantificar as classes de interesse
uni_gen <- unique(dados$info_gen)
N <- length(uni_gen)
# 2. Como executar o procedimento e como acessar os resultados

# 3. Criar uma data.frame (ou tibble) para receber os resultados
coefs <- tibble(info_gen = character(N),
                b0 = numeric(N),
                b1 = numeric(N))
plot(dados$idade, dados$volume, xlab = "Idade (anos)",
     ylab = "Volume (m³/ha)")
cores <- c("orange", "green", "red")
# 4. Executar o processo de repetição (loop)
for (i in 1:N) {
   #     4.1. Filtrar a classe de interesse
   d_i <- subset(dados, info_gen == uni_gen[i])
   #     4.2. Executar os comandos necessários
   nls_i <- nlsLM(volume ~ exp(b0 - b1/idade), d_i,
                  start = list(b0 = 1, b1 = 1))
   points(d_i$idade, fitted(nls_i), col = cores[i])
   #     4.3. Acessar os resultados
   resumo_i <- summary(nls_i)
   b0coef = resumo_i$coefficients[, "Estimate"][[1]]
   b1coef = resumo_i$coefficients[, "Estimate"][[2]]
   #     4.4. Modificar a data.frame criada em (3) para receber os resultados
   coefs[i, "info_gen"] <- uni_gen[i]
   coefs[i, "b0"] <- b0coef
   coefs[i, "b1"] <- b1coef
}
legend("topleft", legend = uni_gen, col = cores, pch = 1)
```