Questão 1 (*Tipos de variáveis*) *(2 pontos)*

Crie um data frame com 3 colunas aonde:

1. coluna1: números inteiros de 1 a 3
2. coluna2: números (float) de 1 a 3
3. coluna3: caracteres 1 a 3

(a) Faça a soma da coluna1 com a coluna1. Qual o tipo de resultado? (Inteiro, Float ou caractere)

(b) Faça a soma da coluna1 com a coluna2. Qual o tipo de resultado? (Inteiro, Float ou caractere)

(c) Faça a soma da coluna2 com a coluna3. Qual o tipo de resultado? (Inteiro, Float ou caractere)

Solução:

```
dados <- data.frame(coluna1=c(1L, 2L, 3L),
                    coluna2=c(1,2,3),
                    coluna3=c("1", "2", "3"))
str(dados)


part1 = dados$coluna1 + dados$coluna1
part1
typeof(part1)

part2 = dados$coluna1 + dados$coluna2
part2
typeof(part2)

part3 = dados$coluna1 + dados$coluna3
```

Questão 2 (*Looping*) *(2 pontos)*

uma forma de gerar uma sequência é a função **"seq"**. Seus primeiros dois argumentos são "from" e "to", seguidos por um terceiro, que é "by". Crie uma sequencia do numero 1 ao numero 100, de maneira que a sequencia tenha incrementos de 7 números.

```
minha_sequencia = seq(from=1, to=100, by=7)
```

crie um looping que faz a soma de todos os elementos do vetor "minha_sequencia". Aonde a cada interação ele deve imprimir no console qual a soma atual, qual o número a ser somado, e o resultado da soma.

Solução:

```
soma = 0
for( i in minha_sequencia){
  print(sprintf("soma atual %3.0f", soma))
  print(sprintf("item a ser somado %3.0f", i))

  soma = soma + i
  print(sprintf("Resultado da soma %3.0f", soma))
}
```

Questão 3 (*Navegando no dataframe*) *(2 pontos)*

Primeiramente vamos converter o banco de dados "AirPassengers" para um data.frame.

1. O nome do banco convertido será "data"
2. Vamos nomear a coluna do banco "data" de "Passengers"
3. Vamos criar uma nova coluna chamado "date" com as datas associadas.
4. Vamos converter os dados da coluna "Passengers" para o tipo "numeric"

```
data <- as.data.frame(AirPassengers)
colnames(data) <- "Passengers"
data$date <- seq(from=as.Date("1949-01-01"), to=as.Date("1960-12-01"), by="month")
data$Passengers <-  as.numeric(data$Passengers)
```

Utilizando o banco de dados convertido:

(a) Crie uma coluna chamada "UpDown", iniciada com o valor numérico NA (as.numeric(NA)).

(b) crie um *looping* que em cada linha verifica se o numero de passageiros foi maior ou menor do que o numero de passageiros no período anterior (linha anterior). Se for maior ele deve colocar o valor de +1 na coluna "UpDown", Se for menor ele deve colocar o valor de -1 na coluna "UpDown", Se for igual ele deve colocar o valor de 0 na coluna "UpDown".

(c) Faça um gráfico de barras da variável "UpDown"

Solução:

```r
data$UpDown <- NA
data$UpDown <- as.numeric(NA)
str(data)

for( row in 1: nrow(data) ) {

  if(row == 1) {
    # se é a primeira linha nao existe linha anterior, pulamos entao essa linha
    next()
  }

  # busca os valores da linha atual e linha anterior
  valor_atual <- data$Passengers[row]
  valor_anterior <- data$Passengers[row-1]

  if(valor_atual == valor_anterior){
    data$UpDown[row] <- 0
  } else if (valor_atual > valor_anterior) {
    data$UpDown[row] <- +1
  } else if (valor_atual < valor_anterior) {
    data$UpDown[row] <- -1
  } else {
    print("NAO DEVERIA CHEGAR AQUI. ALGUM ERRO NA ROTINA!")
  }
}

ggplot(data) +
  geom_bar(aes(x=UpDown))
```

Questão 4 (*Funções básicas*) *(2 pontos)*

Utilizando o banco de dados iris, crie uma cópia do banco chamada de data. No banco de dados "data" crie 4 novas variáveis:

(a) variável 1 (ln_Sepal.Length): deve ser o logaritimo neperiano da variável "Sepal.Length"

(b) variável 2 (exp_Sepal.Length): deve ser o exponencial neperiano da variável "Sepal.Length"

(c) variável 3 (std_Sepal.Length): deve seguir a formula:

$$std\_Sepal.Length_i = \frac{Sepal.Length_i - mean(data\$Sepal.Length)}{sd(data\$Sepal.Length)}$$

(d) variável 4 (iris_idx): deve ser a média aritmética das variáveis Sepal Length, Sepal Width, Petal Length, Petal Width;

Solução:

```r
data <- iris

# item 1
data$ln_Sepal.Length = log(data$Sepal.Length)

# item 2
data$exp_Sepal.Length = exp(data$Sepal.Length)

# item 3
data$std_Sepal.Length = (data$Sepal.Length -
                         mean(data$Sepal.Length))/sd(data$Sepal.Length)

# item 4 (metodo 1)
data$iris_idx = (data$Sepal.Length +
                 data$Sepal.Width +
                 data$Petal.Length +
                 data$Petal.Width)/4

# item 4 (metodo 2)
for(row in 1:nrow(data)){
  data$iris_idx[row] =  mean(as.numeric(data[row,1:4]))
}

# item 4 (metodo 3)
data$iris_idx = rowMeans(data[, 1:4])

# item 4 (metodo 4)
data$iris_idx =  apply(X = data[, 1:4], FUN = mean, MARGIN = 1)
```

Questão 5 (*break*) *(2 pontos)*

Faça um looping de 10.000 interações com as seguintes intruções de execução:

1. sorteie um número aleatorio entre zero e um utilizando a função runif(1);
2. verifique se o número sorteado é maior que 0.8, em caso afirmativo ele deve parar no meio utilizando o comando break; caso contrário deve continuar o processo.

Solução:

```r
contador <- 1
while(contador <= 10000){
  # sorteio do numero
  numSorteado <- runif(1)

  if(numSorteado > 0.8){
    cat(sprintf("numero sorteado (%f) maior que 0.8. (interação %d)",
    numSorteado, contador))
    break;
  }

  # Incremento do contador
  contador = contador+1
}
```