Lista 1 - Introdução a Análise de Dados
Funções
Gabarito

Guilherme Masuko

March 2023

Questão 1

Crie uma função que recebe três parâmetros a, b, c, os coeficientes de uma função do segundo grau e retorne as raízes dessa função.

Uma função do segundo grau tem a seguinte forma

$$f(x) = ax^2 + bx + c$$

As raízes de uma função do segundo grau são os x's cujo a função $f(x)$ cruza o eixo das abscissas (horizontal), isso é, $f(x) = 0$.

Para criar a função precisamos relembrar da fórmula de Bhaskara.[1]

$$\Delta = b^2 - 4 \cdot a \cdot c$$
$$x_{1,2} = \frac{-b \pm \sqrt{\Delta}}{2 \cdot a}$$

Lembre-se dos casos em que Δ é negativo, positivo e igual a zero.

Teste sua função para as seguintes funções:

- $f(x) = x^2$
- $f(x) = 2x^2 - 18$
- $f(x) = x^2 - 4x + 10$

[1] <https://pt.wikipedia.org/wiki/F%C3%B3rmula_quadr%C3%A1tica>

- $f(x) = -2x^2 + 20x - 50$

Solução

- $f(x) = x^2$

$$\Delta = 0^2 - 4 \cdot 1 \cdot 0$$
$$x_{1,2} = \frac{-0}{2 \cdot 1} = 0$$

- $f(x) = 2x^2 - 18$

$$\begin{aligned}\Delta &= 0^2 - 4 \cdot 2 \cdot (-18) \\ &= 144 \\ x_1 &= \frac{-0 + \sqrt{144}}{2 \cdot 2} \\ &= \frac{12}{4} = 3 \\ x_2 &= \frac{-0 - \sqrt{144}}{2 \cdot 2} \\ &= \frac{-12}{4} = -3\end{aligned}$$

- $f(x) = x^2 - 4x + 10$

$$\begin{aligned}\Delta &= (-4)^2 - 4 \cdot 1 \cdot 10 \\ &= 16 - 40 = -24 \\ \Delta &< 0\end{aligned}$$

- $f(x) = -2x^2 + 20x - 50$

$$\begin{aligned}\Delta &= 20^2 - 4 \cdot (-2) \cdot (-50) \\ &= 400 - 400 = 0 \\ x_{1,2} &= \frac{-20}{2 \cdot (-2)} \\ &= \frac{-20}{-4} = 5\end{aligned}$$

```
raizes_funcao_2grau <- function(a, b, c){
  # primeiro definimos o delta
```

```
  delta = b^2 - 4*a*c
  # apos isso, vamos quebrar nos três casos
  if(delta < 0){
    # se o delta é negativo, a parabola não tem raíz
    return(cat("Essa função não tem raízes. A parábola não corta o
       eixo das abscissas!"))
  } else if(delta == 0){
    # se o delta é igual a zero, a raíz é unitária
    x = (-b)/(2*a)
    return(cat("Essa função tem apenas uma raíz. Seu valor é",x,"."))
  } else{
    # se o delta é positivo, então temos duas raízes
    x_1 = (-b + sqrt(delta))/(2*a)
    x_2 = (-b - sqrt(delta))/(2*a)
    return(cat("Essa função tem duas raízes. As duas raízes dessa
       função são respectivamente",x_1,"e",x_2,"."))
  }
}



# Testando nas funcoes

# f(x) = x^2
raizes_funcao_2grau(1, 0, 0)


# f(x) = 2x^2 - 18
raizes_funcao_2grau(a=2, b=0, c=-18)


# f(x) = x^2 - 4x + 10
raizes_funcao_2grau(a=1, b=-4, c=10)


# f(x) = -2x^2 + 20x - 50
raizes_funcao_2grau(a=-2, b=20, c=-50)
```

Questão 2

Com base na questão anterior, crie uma nova função que recebe os mesmos parâmet-

ros a, b e c, coeficientes de uma função do segundo grau, mas que agora retorno um gráfico dessa função.

Solução

```
grafico_funcao_2grau <- function(a, b, c) {
  # primeiro geramos uma sequência de valores para x, de -10 a 10,
     de meio em meio
  x <- seq(-10, 10, 0.5)
  # depois geramos uma sequência de valores nulos do mesmo tamanho
     que o vetor x
  y <- rep(0, length(x))
  # precisamos de uma variável de indice
  index <- 0
  # agora faremos um loop, para cada valor de x, temos um valor de y
     respectivo
  for (valor_x in x) {
    index <- index + 1
    y[index] <- a * valor_x^2 + b * valor_x + c
  }
  plot(x, y)
}

# f(x) = x^2
grafico_funcao_2grau(1, 0, 0)


# f(x) = 2x^2 - 18
grafico_funcao_2grau(a=2, b=0, c=-18)


# f(x) = x^2 - 4x + 10
grafico_funcao_2grau(a=1, b=-4, c=10)


# f(x) = -2x^2 + 20x - 50
grafico_funcao_2grau(a=-2, b=20, c=-50)
```

Plots resultantes.

Figure 1: $f(x) = x^2$

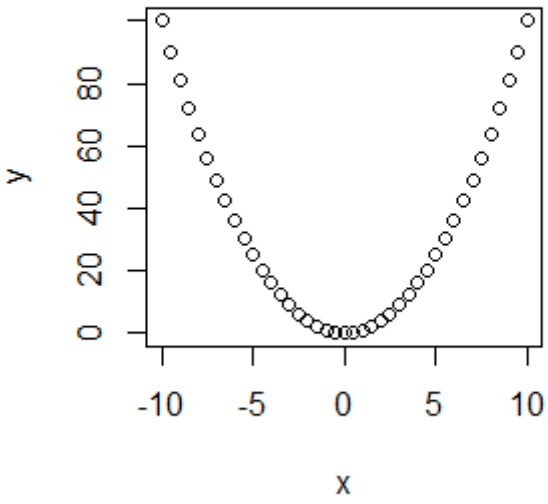


Figure 2: $f(x) = 2x^2 - 18$

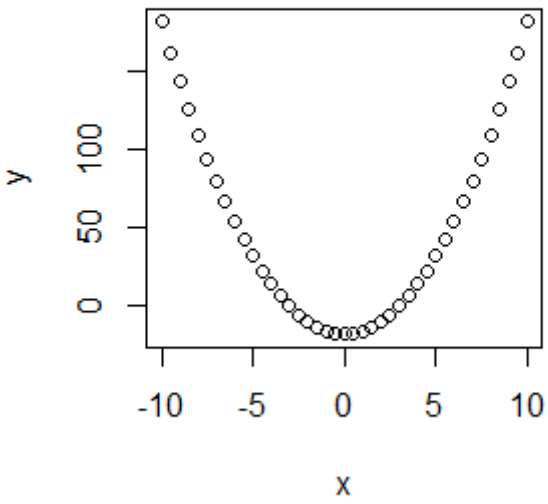


Figure 3: $f(x) = x^2 - 4x + 10$

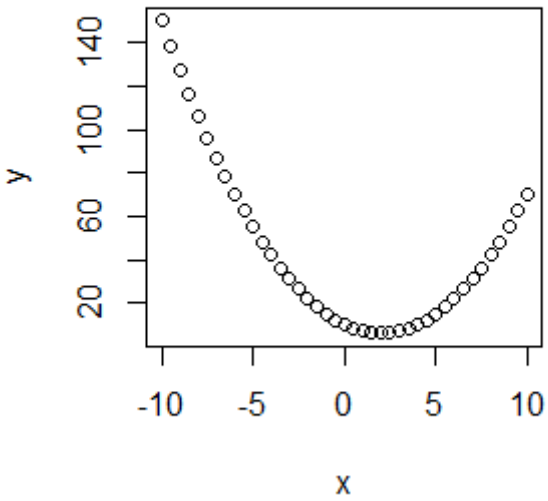
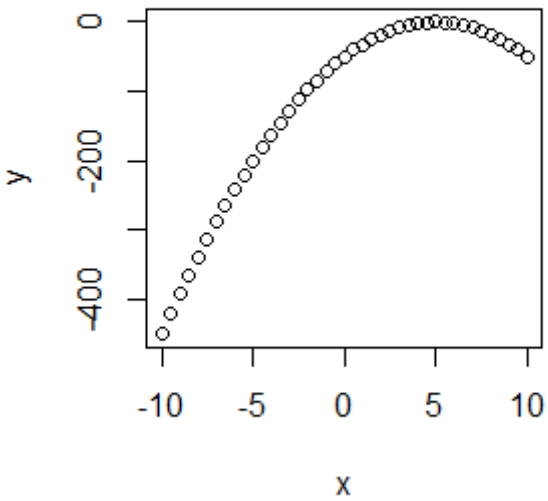


Figure 4: $f(x) = -2x^2 + 20x - 50$



Questão 3

Crie uma função que recebe um número como parâmetro e devolve se o número é primo ou não.

Definição de número primo: Um número primo é um número natural maior que um, tal que não é resultado do produto da multiplicação de dois números naturais, isso é, são números que são divisíveis apenas por ele mesmo dentro da classe dos naturais sem o 1.[2]

Dica: A função que retorna o resto da divisão inteira é '%%'.

Teste sua função para os seguintes valores

- valores de 2 à 20
- 577
- 753
- 997

Solução

```
numero_primo <- function(numero) {
  # precisamos criar uma sequência na qual um loop será usado
  seq = 2:(numero-1)
  # criamos também uma variável lógica (binária) que recebe TRUE (se
     for primo)
  # e FALSE (caso contrário)
  e_primo <- TRUE
  if (numero == 2) {
    # a sequência para o número 2 ficaria 2:1, não é a maneira que
       gostaríamos,
    # para esse caso sabemos que o 2 é um número primo
    return(paste(numero,'é um número primo'))
  } else {
    # caso contrário, fazemos um loop, se o parâmetro passado for
       divisível por
    # algum valor dentre os pertencentes a sequência, então não é
       primo
    for (i in seq) {
      if ((numero %% i) == 0 ) {
        e_primo <- FALSE
        # se for divisível, guardaremos os valores aos quais decompõe
           o parâmetro
        decomp_1 <- i
```

[2]<https://pt.wikipedia.org/wiki/Numero_primo>

```
      decomp_2 <- (numero / i)
      # break é uma função que força o loop a parar
      break
    }
  }
  if (e_primo == TRUE) {
    return(paste(numero,'é um número primo'))
  } else {
    return(paste(numero,'não é um número primo
       porque',decomp_1,'x',decomp_2,'=',numero))
  }
 }
}



# valores de 2 à 20

numero_primo(2)
numero_primo(3)
numero_primo(4)
numero_primo(5)
numero_primo(6)
numero_primo(7)
numero_primo(8)
numero_primo(9)
numero_primo(10)
numero_primo(11)
numero_primo(12)
numero_primo(13)
numero_primo(14)
numero_primo(15)
numero_primo(16)
numero_primo(17)
numero_primo(18)
numero_primo(19)
numero_primo(20)


# 577
```

```
numero_primo(577)

# 753
numero_primo(753)

# 997
numero_primo(997)
```