Guía de ejercicios

Uriel Paluch

7/9/2021

Métodos:

```
#Método de Bisección
Biseccion <- function(a, b, N = 100, tol) {
#Tiene por default 100 iteraciones

  #Instancio las listas vacias
  lista_a <- c(NULL)
  lista_b <- c(NULL)
  lista_p <- c(NULL)

  for (i in 1:N) {
    #Calculo P
    p <- (a+b)/2

    #Agrego el valor a cada lista
    lista_p[i] <- p
    lista_a[i] <- a
    lista_b[i] <- b

    #Evaluo la función en p
    fp <- f(p)

    #Si la f(p) es 0, entonces es raiz
    #O si esta dentro del límite tolerado
    if (fp == 0 | abs((b-a)/2) <= tol) {
      #Creo un data frame con las listas
      datos <- data.frame(lista_a, lista_b, lista_p)
      colnames(datos) <- c("A", "B", "P")
      print(datos)
      return(paste("La raiz es: ", p))
    }

    #Si comparten el mismo signo
    if (fp * f(a) > 0) {
      a <- p
    } else {
      b <- p
    }
  }

  #En el caso de que falle el método
```

```r
  return(paste('El método falla luego de: ', N, ' iteraciones'))
}
```

```r
#Método de punto fijo
PuntoFijo <- function(p0, n = 100, tol){
  #Donde p0 es la aproximación inicial
  #El número máximo de iteraciones n viene por default en 100
  #Y tol es la toleranacia al error

  #Instancio las listas vacias
  lista_p <- c(NULL)
  lista_gp <- c(NULL)

  for (i in 1:n) {
    #Calculo p
    p <- f(p0)

    lista_p[i] <- p0
    lista_gp[i] <- p

    if(abs(p-p0) <= tol){
      #Creo un data frame con las listas
      datos <- data.frame(lista_p, lista_gp)
      colnames(datos) <- c("P", "G(P)")
      print(datos)
      return(p)
    }

    p0 <- p
  }

  #En el caso de que falle el método
  return(paste('El método falla luego de: ', n, ' iteraciones'))
}
```

```r
Newton <- function(p0, tol, n = 100){
  #Donde p0 es la aproximación inicial
  #El número máximo de iteraciones n viene por default en 100
  #Y tol es la toleranacia al error

  for (i in 1:n) {

    #Calculo p
    p <- p0 - (f(p0)/fprima(p0))

    if(abs(p-p0) <= tol){
      return(p)
    }

    p0 <- p

  }

  #En el caso de que falle el método
```

```r
  return(paste('El método falla luego de: ', n, ' iteraciones'))
}
```

```r
Secante <- function(p0, p1, tol, n = 100){

  q0 = f(p0)
  q1 = f(p1)

  for (i in 2:n) {

    p = p1 - q1 * (p1 - p0) / (q1 - q0)

    if (abs(p-p1) < tol){
      return(p)
    }

    p0 = p1
    q0 = q1
    p1 = p
    q1 = f(p)

  }

  return(paste("El método falló luego de ", n, " iteraciones"))
}
```

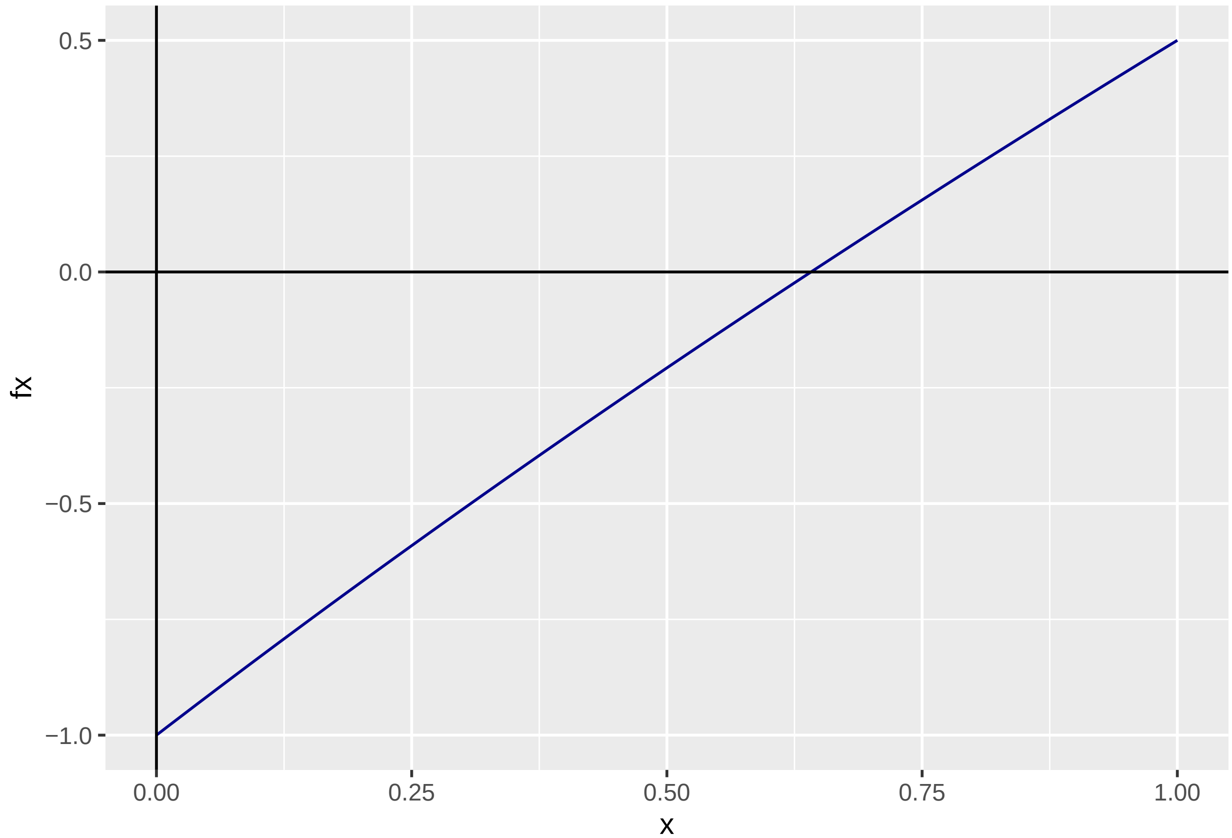
```r
FalsaPosicion <- function(p0, p1, tol, n = 100){

  q0 <- f(p0)
  q1 <- f(p1)

  for (i in 2:n){
    p = p1 -q1*(p1-p0)/(q1-q0)

    if (abs(p-p1) < tol){
      return(p)
    }

    q = f(p)

    if (q*q1 < 0){
      p0 <- p1
      q0 <- q1
    }

    p1 <- p
    q1 <- q
  }

  return(paste("El método falló luego de ", n, " iteraciones"))
}
```

Ejercicio 1:

Aplique el método de Bisección para encontrar soluciones exactas dentro de 0.00001 para los siguientes problemas:

a) $x - 2^(- x) = 0 \qquad 0 \leq x \leq 1$

```
f <- function(x){
  return(x-2^(-x))
}

#Instancio un vector que me va a indicar los puntos en la función
x <- seq(0, 1, by = 0.01)

#Genero los puntos
fx <- f(x)

#Creo un data frame con los x e y
df <- data.frame(x, fx)

#Instancio los datos
gg_fx <- ggplot(data = df)

#Agrego la capa con los datos
gg_fx <- gg_fx + aes(x = x, y = fx)

#Est grafica una linea
gg_fx <- gg_fx + geom_line(linetype = 1, colour = "darkblue")

#Agrego el eje X
gg_fx <- gg_fx + geom_vline(xintercept = 0, linetype = 1)

#Agrego el eje Y
gg_fx <- gg_fx + geom_hline(yintercept = 0, linetype = 1)

#Grafico
gg_fx
```




```
Biseccion(a = 0.5,b = 1, tol = 0.00001)
```

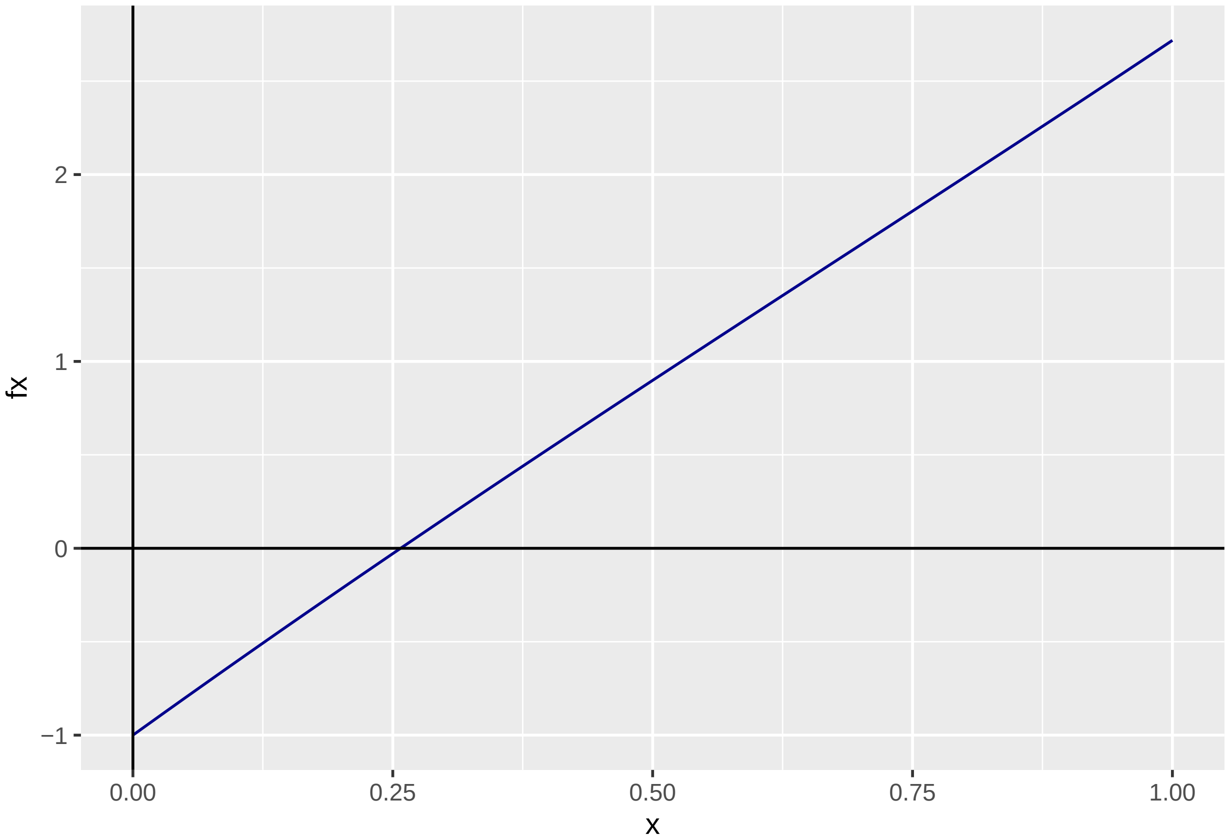
```
##            A         B         P
## 1  0.5000000 1.0000000 0.7500000
## 2  0.5000000 0.7500000 0.6250000
## 3  0.6250000 0.7500000 0.6875000
## 4  0.6250000 0.6875000 0.6562500
## 5  0.6250000 0.6562500 0.6406250
## 6  0.6406250 0.6562500 0.6484375
## 7  0.6406250 0.6484375 0.6445312
## 8  0.6406250 0.6445312 0.6425781
## 9  0.6406250 0.6425781 0.6416016
## 10 0.6406250 0.6416016 0.6411133
## 11 0.6411133 0.6416016 0.6413574
## 12 0.6411133 0.6413574 0.6412354
## 13 0.6411133 0.6412354 0.6411743
## 14 0.6411743 0.6412354 0.6412048
## 15 0.6411743 0.6412048 0.6411896
## 16 0.6411743 0.6411896 0.6411819
```

```
## [1] "La raiz es:  0.641181945800781"
```

b) $e^x - x^2 + 3 * x - 2 = 0 \qquad 0 \le x \le 1$

```
f <- function(x){
  return(exp(x) - x^2+3*x - 2)
}
```

```r
#Instancio un vector que me va a indicar los puntos en la función
x <- seq(0, 1, by = 0.01)

#Genero los puntos
fx <- f(x)

#Creo un data frame con los x e y
df <- data.frame(x, fx)

#Instancio los datos
gg_fx <- ggplot(data = df)

#Agrego la capa con los datos
gg_fx <- gg_fx + aes(x = x, y = fx)

#Est grafica una linea
gg_fx <- gg_fx + geom_line(linetype = 1, colour = "darkblue")

#Agrego el eje X
gg_fx <- gg_fx + geom_vline(xintercept = 0, linetype = 1)

#Agrego el eje Y
gg_fx <- gg_fx + geom_hline(yintercept = 0, linetype = 1)

#Grafico
gg_fx
```




```
Biseccion(a = 0,b = 0.5, tol = 0.00001)
```

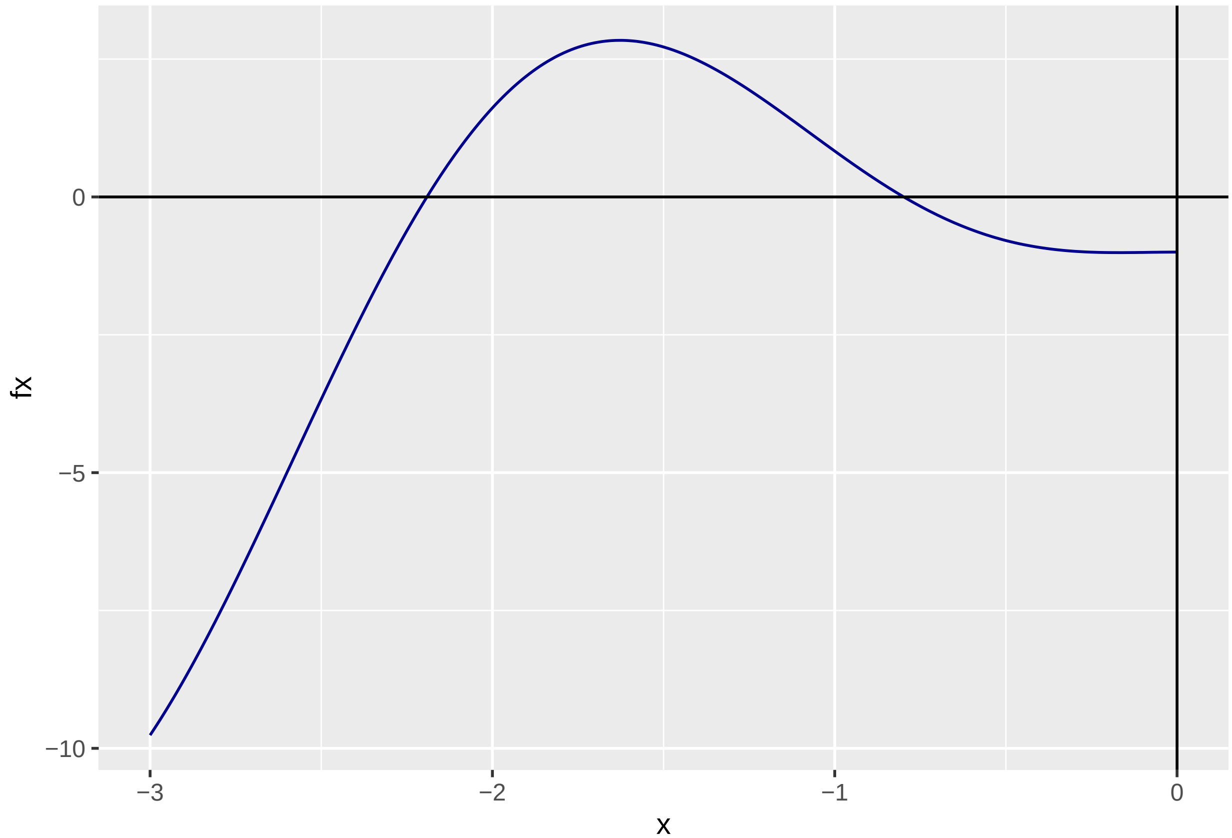
```
##            A         B         P
## 1  0.0000000 0.5000000 0.2500000
## 2  0.2500000 0.5000000 0.3750000
## 3  0.2500000 0.3750000 0.3125000
## 4  0.2500000 0.3125000 0.2812500
## 5  0.2500000 0.2812500 0.2656250
## 6  0.2500000 0.2656250 0.2578125
## 7  0.2500000 0.2578125 0.2539062
## 8  0.2539062 0.2578125 0.2558594
## 9  0.2558594 0.2578125 0.2568359
## 10 0.2568359 0.2578125 0.2573242
## 11 0.2573242 0.2578125 0.2575684
## 12 0.2573242 0.2575684 0.2574463
## 13 0.2574463 0.2575684 0.2575073
## 14 0.2575073 0.2575684 0.2575378
## 15 0.2575073 0.2575378 0.2575226
## 16 0.2575226 0.2575378 0.2575302
```

```
## [1] "La raiz es:  0.257530212402344"
```

c) $2x * cos(2x) - (x+1)^2 = 0 \qquad -3 \leq x \leq -2 \ -1 \leq x \leq 0$

```
f <- function(x){
  return(2*x*cos(2*x)-(x+1)^2)
}
```

```r
#Instancio un vector que me va a indicar los puntos en la función
x <- seq(-3, 0, by = 0.01)

#Genero los puntos
fx <- f(x)

#Creo un data frame con los x e y
df <- data.frame(x, fx)

#Instancio los datos
gg_fx <- ggplot(data = df)

#Agrego la capa con los datos
gg_fx <- gg_fx + aes(x = x, y = fx)

#Est grafica una linea
gg_fx <- gg_fx + geom_line(linetype = 1, colour = "darkblue")

#Agrego el eje X
gg_fx <- gg_fx + geom_vline(xintercept = 0, linetype = 1)

#Agrego el eje Y
gg_fx <- gg_fx + geom_hline(yintercept = 0, linetype = 1)

#Grafico
gg_fx
```




```
Biseccion(a = -3,b = -2, tol = 0.00001)
```

```
##             A         B         P
## 1  -3.000000 -2.000000 -2.500000
## 2  -2.500000 -2.000000 -2.250000
## 3  -2.250000 -2.000000 -2.125000
## 4  -2.250000 -2.125000 -2.187500
## 5  -2.250000 -2.187500 -2.218750
## 6  -2.218750 -2.187500 -2.203125
## 7  -2.203125 -2.187500 -2.195312
## 8  -2.195312 -2.187500 -2.191406
## 9  -2.191406 -2.187500 -2.189453
## 10 -2.191406 -2.189453 -2.190430
## 11 -2.191406 -2.190430 -2.190918
## 12 -2.191406 -2.190918 -2.191162
## 13 -2.191406 -2.191162 -2.191284
## 14 -2.191406 -2.191284 -2.191345
## 15 -2.191345 -2.191284 -2.191315
## 16 -2.191315 -2.191284 -2.191299
## 17 -2.191315 -2.191299 -2.191307
```

```
## [1] "La raiz es:  -2.19130706787109"
```

```
Biseccion(a = -1,b = 0, tol = 0.00001)
```

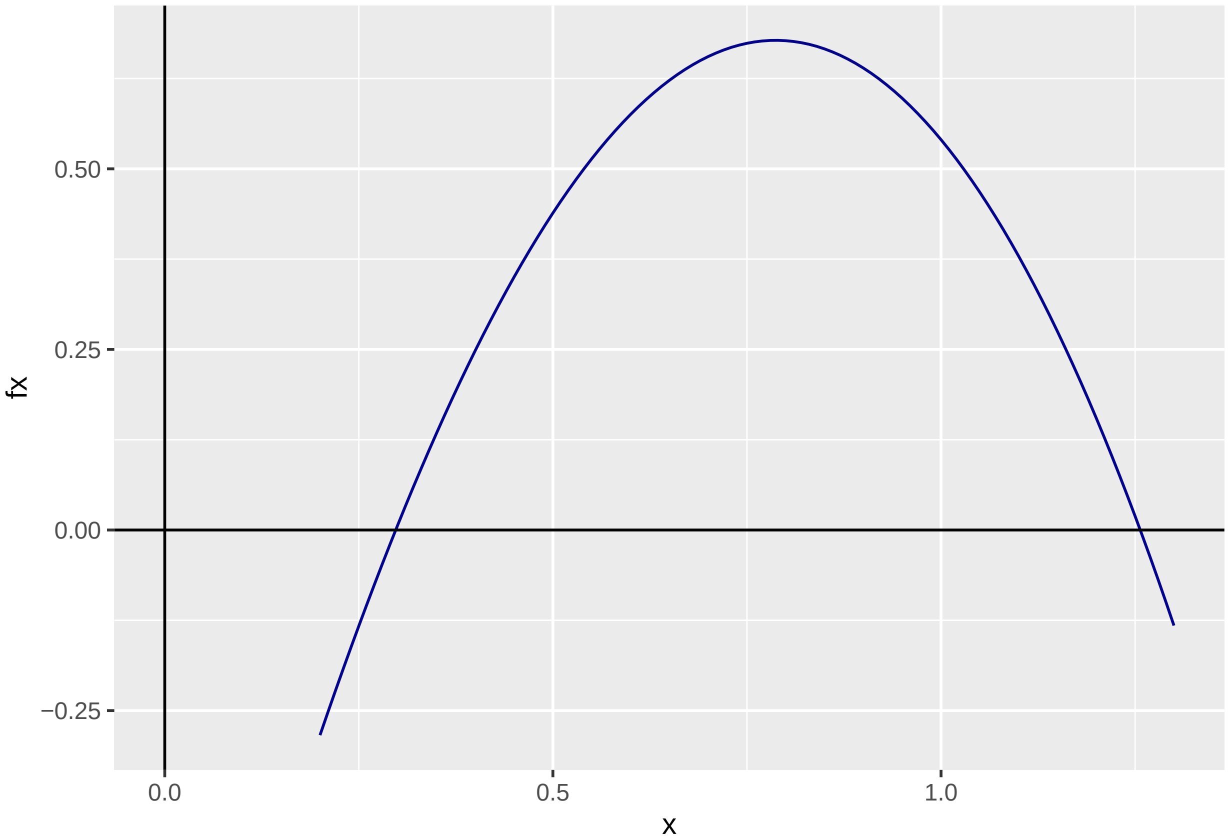
```
##              A          B          P
## 1  -1.0000000  0.0000000 -0.5000000
## 2  -1.0000000 -0.5000000 -0.7500000
```

```
## 3  -1.0000000 -0.7500000 -0.8750000
## 4  -0.8750000 -0.7500000 -0.8125000
## 5  -0.8125000 -0.7500000 -0.7812500
## 6  -0.8125000 -0.7812500 -0.7968750
## 7  -0.8125000 -0.7968750 -0.8046875
## 8  -0.8046875 -0.7968750 -0.8007812
## 9  -0.8007812 -0.7968750 -0.7988281
## 10 -0.7988281 -0.7968750 -0.7978516
## 11 -0.7988281 -0.7978516 -0.7983398
## 12 -0.7983398 -0.7978516 -0.7980957
## 13 -0.7983398 -0.7980957 -0.7982178
## 14 -0.7982178 -0.7980957 -0.7981567
## 15 -0.7982178 -0.7981567 -0.7981873
## 16 -0.7981873 -0.7981567 -0.7981720
## 17 -0.7981720 -0.7981567 -0.7981644
```

```
## [1] "La raiz es:  -0.798164367675781"
```

d) $xcos(x) - 2x^2 + 3x - 1 = 0 \qquad 0.2 \leq x \leq 0.3 \; 1.2 \leq x \leq 1.3$

```
f <- function(x){
  return(x*cos(x)-2*x^2+3*x-1)
}

#Instancio un vector que me va a indicar los puntos en la función
x <- seq(0.2, 1.3, by = 0.01)

#Genero los puntos
fx <- f(x)

#Creo un data frame con los x e y
df <- data.frame(x, fx)

#Instancio los datos
gg_fx <- ggplot(data = df)

#Agrego la capa con los datos
gg_fx <- gg_fx + aes(x = x, y = fx)

#Est grafica una linea
gg_fx <- gg_fx + geom_line(linetype = 1, colour = "darkblue")

#Agrego el eje X
gg_fx <- gg_fx + geom_vline(xintercept = 0, linetype = 1)

#Agrego el eje Y
gg_fx <- gg_fx + geom_hline(yintercept = 0, linetype = 1)

#Grafico
gg_fx
```




```r
Biseccion(a = 0.25,b = 0.5, tol = 0.00001)
```

```
##            A         B         P
## 1  0.2500000 0.5000000 0.3750000
## 2  0.2500000 0.3750000 0.3125000
## 3  0.2500000 0.3125000 0.2812500
## 4  0.2812500 0.3125000 0.2968750
## 5  0.2968750 0.3125000 0.3046875
## 6  0.2968750 0.3046875 0.3007812
## 7  0.2968750 0.3007812 0.2988281
## 8  0.2968750 0.2988281 0.2978516
## 9  0.2968750 0.2978516 0.2973633
## 10 0.2973633 0.2978516 0.2976074
## 11 0.2973633 0.2976074 0.2974854
## 12 0.2974854 0.2976074 0.2975464
## 13 0.2974854 0.2975464 0.2975159
## 14 0.2975159 0.2975464 0.2975311
## 15 0.2975159 0.2975311 0.2975235
```

```
## [1] "La raiz es:  0.297523498535156"
```

```r
Biseccion(a = 1,b = 1.5, tol = 0.00001)
```

```
##           A        B        P
## 1  1.000000 1.500000 1.250000
## 2  1.250000 1.500000 1.375000
## 3  1.250000 1.375000 1.312500
## 4  1.250000 1.312500 1.281250
```

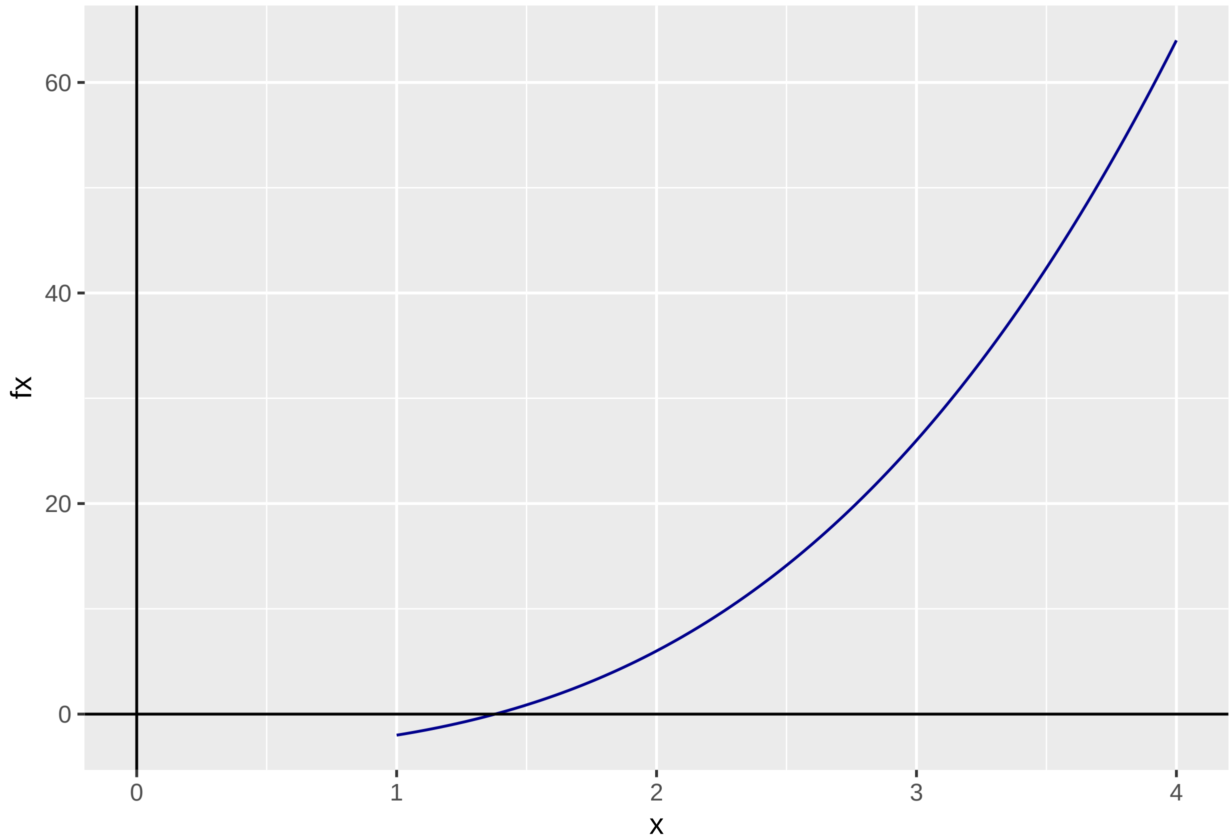
```
## 5  1.250000 1.281250 1.265625
## 6  1.250000 1.265625 1.257812
## 7  1.250000 1.257812 1.253906
## 8  1.253906 1.257812 1.255859
## 9  1.255859 1.257812 1.256836
## 10 1.255859 1.256836 1.256348
## 11 1.256348 1.256836 1.256592
## 12 1.256592 1.256836 1.256714
## 13 1.256592 1.256714 1.256653
## 14 1.256592 1.256653 1.256622
## 15 1.256622 1.256653 1.256638
## 16 1.256622 1.256638 1.256630
```

```
## [1] "La raiz es:  1.25662994384766"
```

Ejercicio 2:

Use el Teorema 2.1 para obtener una cota del número de iteraciones que se requieren para alcanzar una aproximación con una exactitud de 0.001 a la solución de $x^3 + x - 4$ que se encuentra en el intervalo $[1; 4]$. Obtenga una aproximación de la raíz con este grado de exactitud.

```r
f <- function(x){
  return(x^3+x-4)
}

#Instancio un vector que me va a indicar los puntos en la función
x <- seq(1, 4, by = 0.01)

#Genero los puntos
fx <- f(x)

#Creo un data frame con los x e y
df <- data.frame(x, fx)

#Instancio los datos
gg_fx <- ggplot(data = df)

#Agrego la capa con los datos
gg_fx <- gg_fx + aes(x = x, y = fx)

#Est grafica una linea
gg_fx <- gg_fx + geom_line(linetype = 1, colour = "darkblue")

#Agrego el eje X
gg_fx <- gg_fx + geom_vline(xintercept = 0, linetype = 1)

#Agrego el eje Y
gg_fx <- gg_fx + geom_hline(yintercept = 0, linetype = 1)

#Grafico
gg_fx
```




```r
Biseccion(a = 1,b = 4, tol = 0.001)
```

```
##           A        B        P
## 1  1.00000 4.000000 2.500000
## 2  1.00000 2.500000 1.750000
## 3  1.00000 1.750000 1.375000
## 4  1.37500 1.750000 1.562500
## 5  1.37500 1.562500 1.468750
## 6  1.37500 1.468750 1.421875
## 7  1.37500 1.421875 1.398438
## 8  1.37500 1.398438 1.386719
## 9  1.37500 1.386719 1.380859
## 10 1.37500 1.380859 1.377930
## 11 1.37793 1.380859 1.379395
## 12 1.37793 1.379395 1.378662
```

```
## [1] "La raiz es:  1.378662109375"
```

Ejercicio 3:

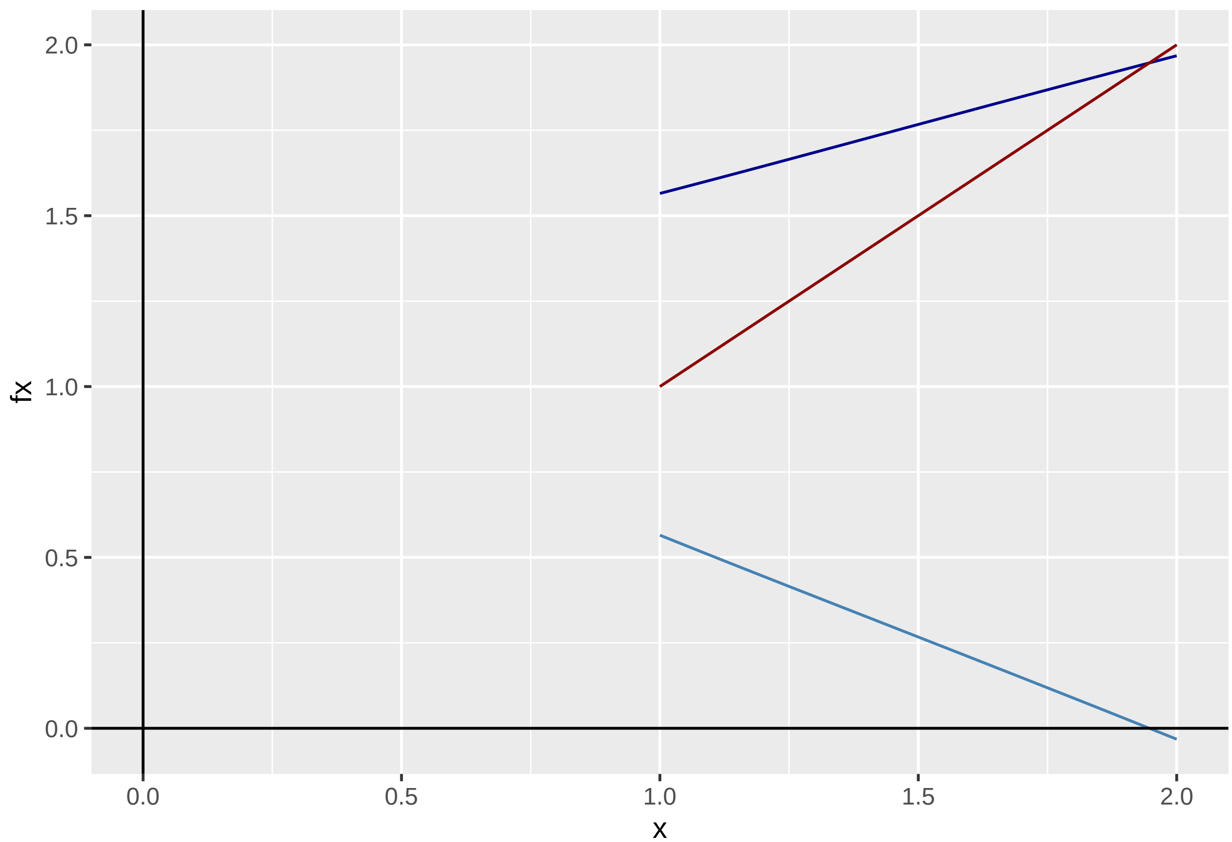
Aplique el método de iteración de punto fijo para determinar una solución con una exactitud de 0.01 para $x^4 - 3x^2 - 3 = 0$ en $[1; 2]$. Utiice $p_0 = 1$.

Reexpreso la ecuación de la forma

$$x = g(x)$$

$x = (3 * x^2 + 3)^{\frac{1}{4}}$

```r
#La función del ejercicio
f <- function(x){
  return((3*x^2+3)^(1/4))
}
#La función para graficar la raiz
g <- function(x){
  return(-x+(3*x^2+3)^(1/4))
}

#Instancio un vector que me va a indicar los puntos en la función
x <- seq(1, 2, by = 0.01)

#Genero los puntos
fx <- f(x)

#Creo un data frame con los x e y
df <- data.frame(x, fx)

#Instancio los datos
gg_fx <- ggplot(data = df)

#Agrego la capa con los datos
gg_fx <- gg_fx + aes(x = x, y = fx)

#Grafica la función del ejercicio
gg_fx <- gg_fx + geom_line(linetype = 1, colour = "darkblue")

#Gráfico x = y
gg_fx <- gg_fx + geom_line(aes(y = x), colour = "darkred")

#Gráfico la función del ejercicio donde esta la raiz
gg_fx <- gg_fx + geom_line(aes(x = x, y = g(x)), colour = "steelblue")

#Agrego el eje X
gg_fx <- gg_fx + geom_vline(xintercept = 0, linetype = 1)

#Agrego el eje Y
gg_fx <- gg_fx + geom_hline(yintercept = 0, linetype = 1)

#Grafico
gg_fx
```



Tomo $p_0 = 1$

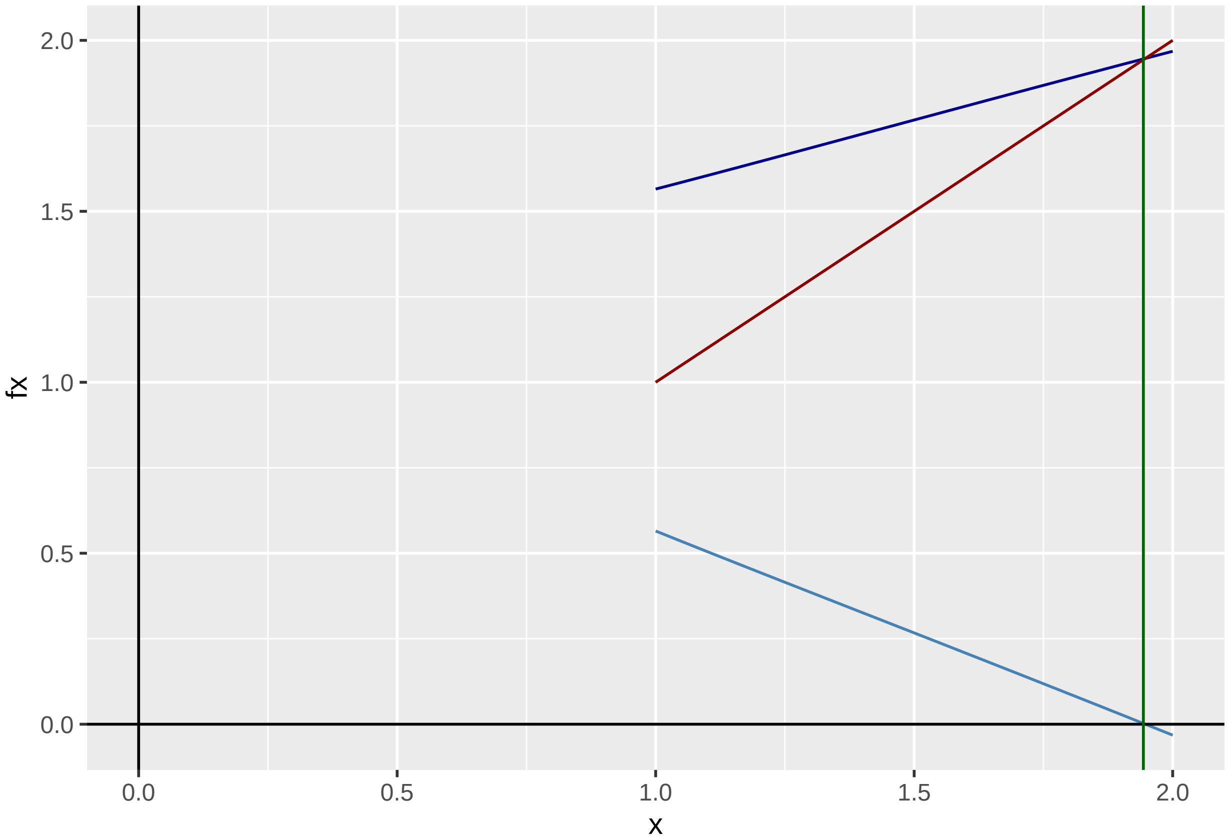
```
this_could_go_wrong <- tryCatch(
  PuntoFijo(p0 = 1, tol = 0.01),
  error = function(e){print("Error")})
```

```
##          P     G(P)
## 1 1.000000 1.565085
## 2 1.565085 1.793573
## 3 1.793573 1.885944
## 4 1.885944 1.922848
## 5 1.922848 1.937508
## 6 1.937508 1.943317
```

```
if (this_could_go_wrong != "Error"){
  raiz <- this_could_go_wrong
  print(paste("La raiz es: ", raiz))
}
```

```
## [1] "La raiz es:  1.94331692989868"
```

```
#Agrego la linea recta que une los puntos entre la función del ejercicio, el punto fijo y la raiz
gg_fx <- gg_fx + geom_vline(xintercept = raiz, linetype = 1, colour="darkgreen")

gg_fx
```



Ejercicio 4:

Aplique el método de iteración de punto fijo para determinar una solución con una exactitud de 0.01 para $x^3 - x - 1 = 0$ en $[1;2]$. Utiice $p_0 = 1$

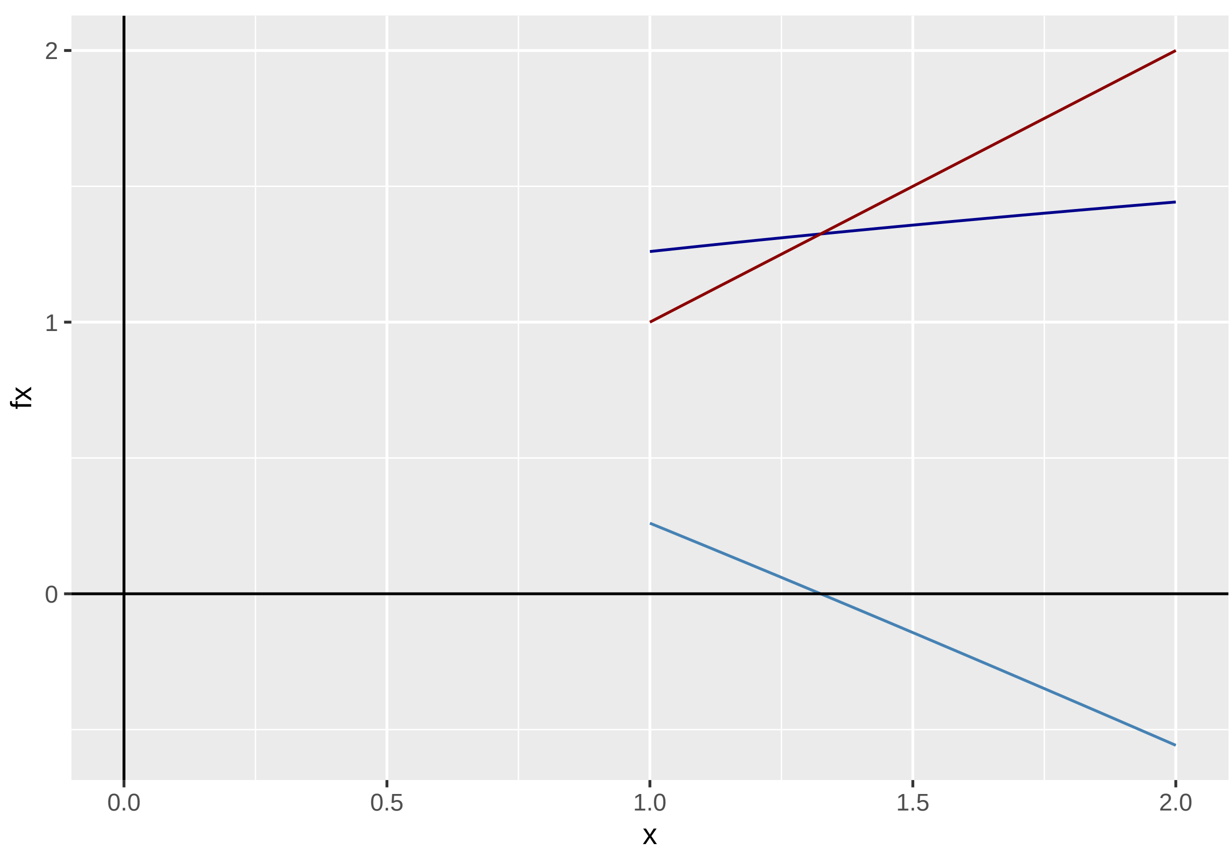
Reexpreso la ecuación de la forma

$$x = g(x)$$

$x = (x+1)^{\frac{1}{3}}$

```
#La función del ejercicio
f <- function(x){
  return((x+1)^(1/3))
}
#La función para graficar la raiz
g <- function(x){
  return(-x+(x+1)^(1/3))
}

#Instancio un vector que me va a indicar los puntos en la función
x <- seq(1, 2, by = 0.01)

#Genero los puntos
fx <- f(x)

#Creo un data frame con los x e y
```

```r
df <- data.frame(x, fx)

#Instancio los datos
gg_fx <- ggplot(data = df)

#Agrego la capa con los datos
gg_fx <- gg_fx + aes(x = x, y = fx)

#Grafica la función del ejercicio
gg_fx <- gg_fx + geom_line(linetype = 1, colour = "darkblue")

#Gráfico x = y
gg_fx <- gg_fx + geom_line(aes(y = x), colour = "darkred")

#Gráfico la función del ejercicio donde esta la raiz
gg_fx <- gg_fx + geom_line(aes(x = x, y = g(x)), colour = "steelblue")

#Agrego el eje X
gg_fx <- gg_fx + geom_vline(xintercept = 0, linetype = 1)

#Agrego el eje Y
gg_fx <- gg_fx + geom_hline(yintercept = 0, linetype = 1)

#Grafico
gg_fx
```



Tomo $p_0 = 1$

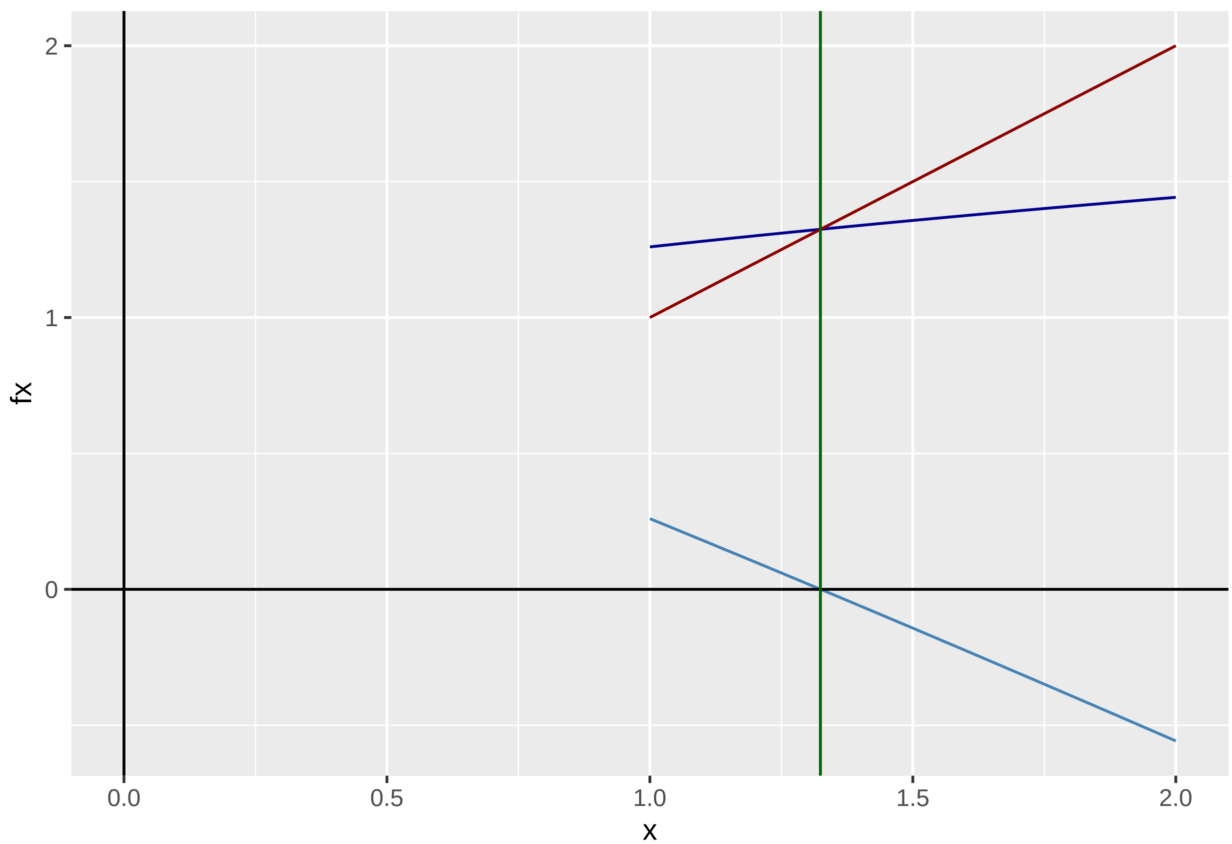
```
this_could_go_wrong <- tryCatch(
  PuntoFijo(p0 = 1, tol = 0.01),
  error = function(e){print("Error")})
```

```
##          P     G(P)
## 1 1.000000 1.259921
## 2 1.259921 1.312294
## 3 1.312294 1.322354
## 4 1.322354 1.324269
```

```
if (this_could_go_wrong != "Error"){
  raiz <- this_could_go_wrong
  print(paste("La raiz es: ", raiz))
}
```

```
## [1] "La raiz es:  1.32426874455158"
```

```
#Agrego la linea recta que une los puntos entre la función del ejercicio, el punto fijo y la raiz
gg_fx <- gg_fx + geom_vline(xintercept = raiz, linetype = 1, colour="darkgreen")

gg_fx
```



Ejercicio 5:

Aplique el Teorema de Punto Fijo para demostrar que $g(x) = \pi + 0.5sen(\frac{x}{2})$ tiene un único punto fijo en $[0; 2 * \pi]$. Use la iteración de punto fijo para obtener una aproximación al punto fijo con una exactitud de

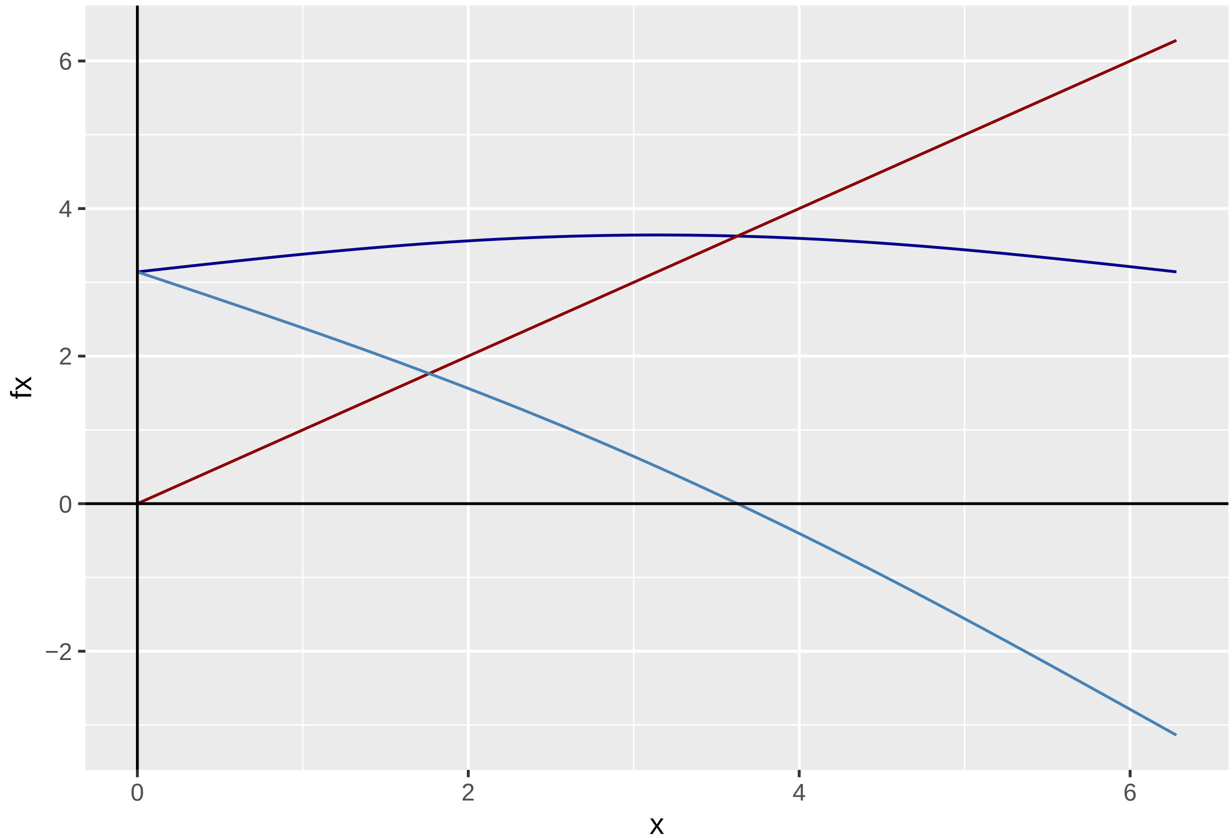
0.01. Use el Corolario para estimar la cantidad de iteraciones necesarias para alcanzar una exactitud de 0.001 y después compare esta estimación teórica con la cantidad que realmente se requiere.

Reexpreso la ecuación de la forma

$$x = g(x)$$

$x = \pi + 0.5 * sen(\frac{x}{2})$

```
#La función del ejercicio
f <- function(x){
  return( pi + 0.5 * sin(x/2))
}
#La función para graficar la raiz
g <- function(x){
  return(-x+ pi + 0.5 * sin(x/2))
}

#Instancio un vector que me va a indicar los puntos en la función
x <- seq(0, 2*pi, by = 0.01)

#Genero los puntos
fx <- f(x)

#Creo un data frame con los x e y
df <- data.frame(x, fx)

#Instancio los datos
gg_fx <- ggplot(data = df)

#Agrego la capa con los datos
gg_fx <- gg_fx + aes(x = x, y = fx)

#Grafica la función del ejercicio
gg_fx <- gg_fx + geom_line(linetype = 1, colour = "darkblue")

#Gráfico x = y
gg_fx <- gg_fx + geom_line(aes(y = x), colour = "darkred")

#Gráfico la función del ejercicio donde esta la raiz
gg_fx <- gg_fx + geom_line(aes(x = x, y = g(x)), colour = "steelblue")

#Agrego el eje X
gg_fx <- gg_fx + geom_vline(xintercept = 0, linetype = 1)

#Agrego el eje Y
gg_fx <- gg_fx + geom_hline(yintercept = 0, linetype = 1)

#Grafico
gg_fx
```



Tomo $p_0 = 3.5$

```
this_could_go_wrong <- tryCatch(
  PuntoFijo(p0 = 3.5, tol = 0.01),
  error = function(e){print("Error")})
```

```
##          P     G(P)
## 1 3.500000 3.633586
## 2 3.633586 3.626540
```

```
if (this_could_go_wrong != "Error"){
  raiz <- this_could_go_wrong
  print(paste("La raiz es: ", raiz))
}
```

```
## [1] "La raiz es:  3.62654022318492"
```

```
this_could_go_wrong <- tryCatch(
  PuntoFijo(p0 = 3.5, tol = 0.0001),
  error = function(e){print("Error")})
```

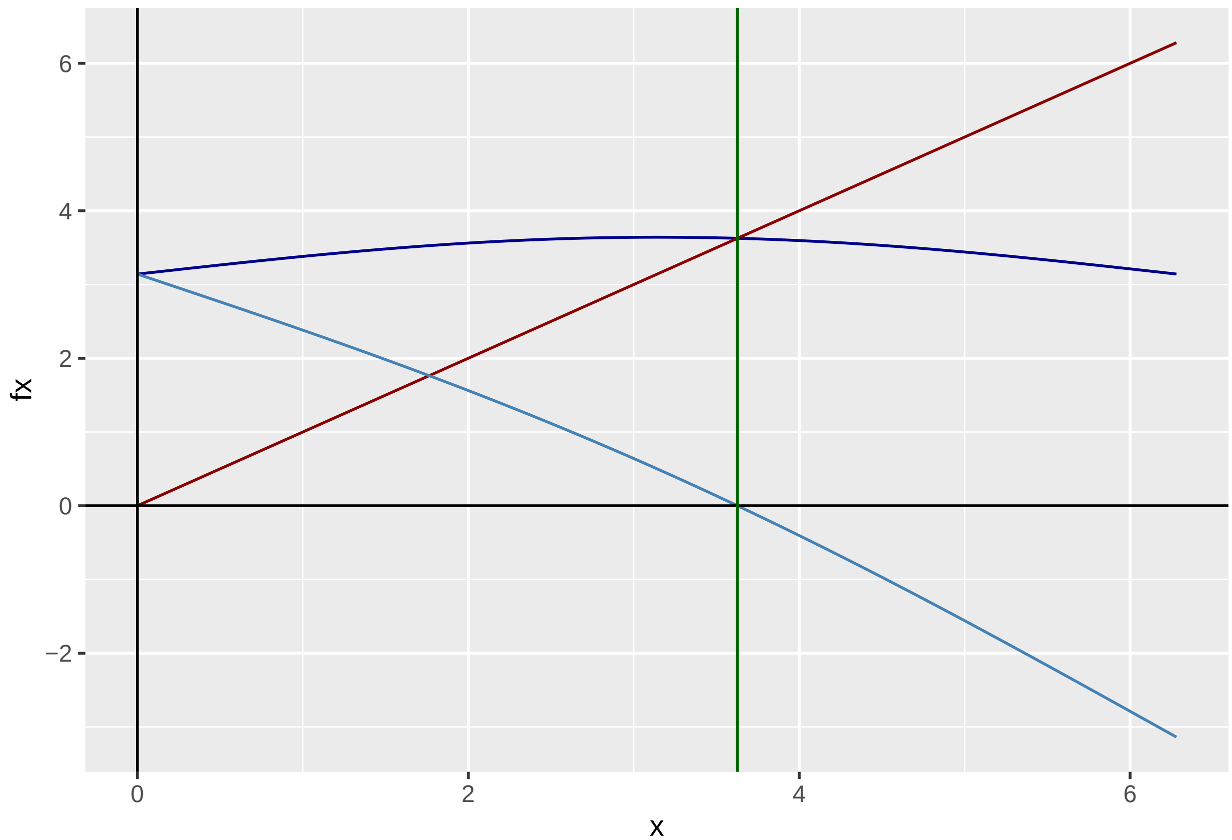
```
##          P     G(P)
## 1 3.500000 3.633586
## 2 3.633586 3.626540
## 3 3.626540 3.626966
## 4 3.626966 3.626941
```

```
if (this_could_go_wrong != "Error"){
  raiz <- this_could_go_wrong
```

```
  print(paste("La raiz es: ", raiz))
}
```

```
## [1] "La raiz es:  3.6269405653587"
```

```
#Agrego la linea recta que une los puntos entre la función del ejercicio, el punto fijo y la raiz
gg_fx <- gg_fx + geom_vline(xintercept = raiz, linetype = 1, colour="darkgreen")

gg_fx
```

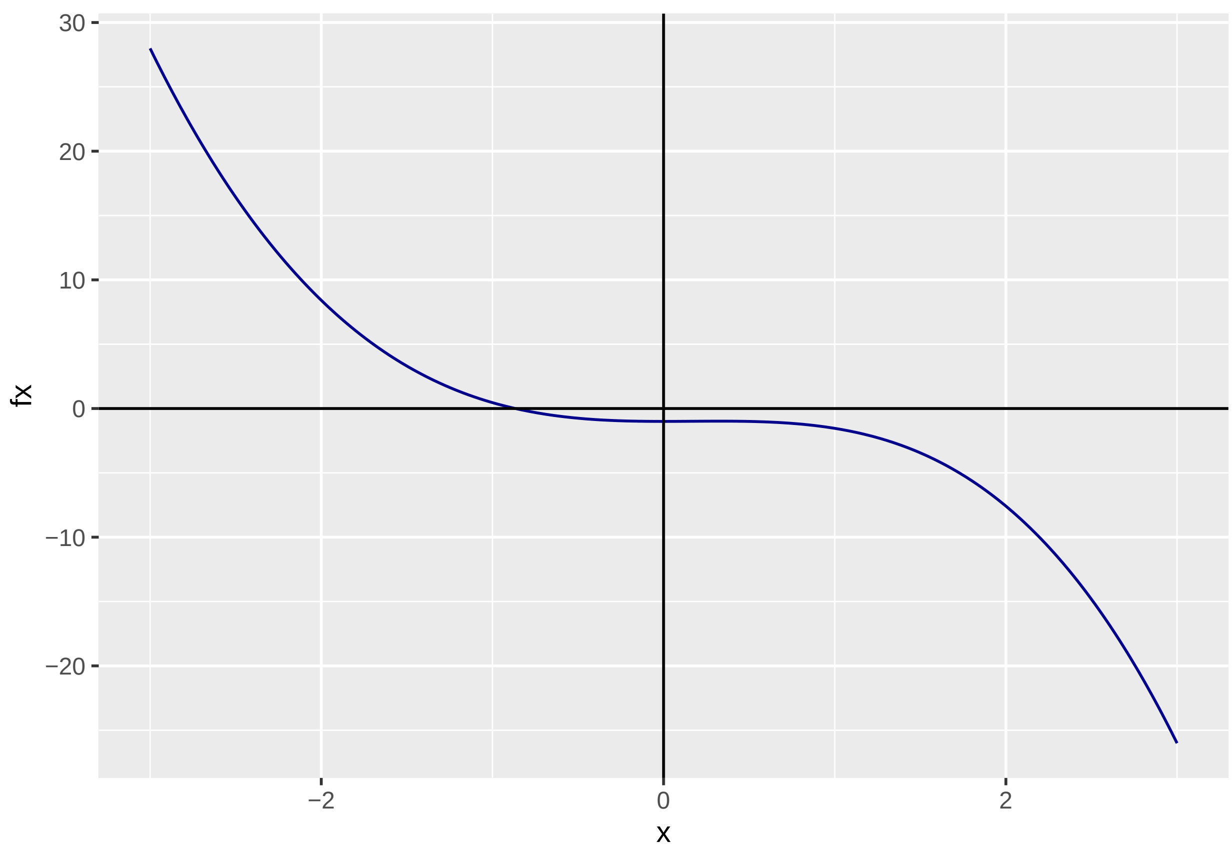


Ejercicio 6:

Sean $f(x) = -x^3 - cos(x)$ y $p_0 = 1$. Aplique la fórmula de iteración deNewton para encontrar p_2. ¿Podríamos utilizar $p_0 = 0$?

```
f <- function(x){
  return(-x^3 -cos(x))
}

#Instancio un vector que me va a indicar los puntos en la función
x <- seq(-3, 3, by = 0.01)

#Genero los puntos
fx <- f(x)

#Creo un data frame con los x e y
df <- data.frame(x, fx)
```

```r
#Instancio los datos
gg_fx <- ggplot(data = df)

#Agrego la capa con los datos
gg_fx <- gg_fx + aes(x = x, y = fx)

#Est grafica una linea
gg_fx <- gg_fx + geom_line(linetype = 1, colour = "darkblue")

#Agrego el eje X
gg_fx <- gg_fx + geom_vline(xintercept = 0, linetype = 1)

#Agrego el eje Y
gg_fx <- gg_fx + geom_hline(yintercept = 0, linetype = 1)

#Grafico
gg_fx
```




```r
fprima <- D(expression(-x^3 -cos(x)), "x")

fprima
```

```
## -(3 * x^2 - sin(x))
```

```r
fprima <- function(x){
  return(-(3 * x^2 - sin(x)))
```

```r
}

Newton(p0 = -1, tol = 0.001)
```

```
## [1] -0.8654741
```

```r
this_could_go_wrong <- tryCatch(
  Newton(p0 = 0, tol = 0.001),
  error = function(e){print("Error")})
```

```
## Warning in cos(x): Se han producido NaNs

## Warning in sin(x): Se han producido NaNs

## [1] "Error"
```

```r
if (this_could_go_wrong != "Error"){
  raiz <- this_could_go_wrong
  print(paste("La raiz es: ", raiz))
}
```

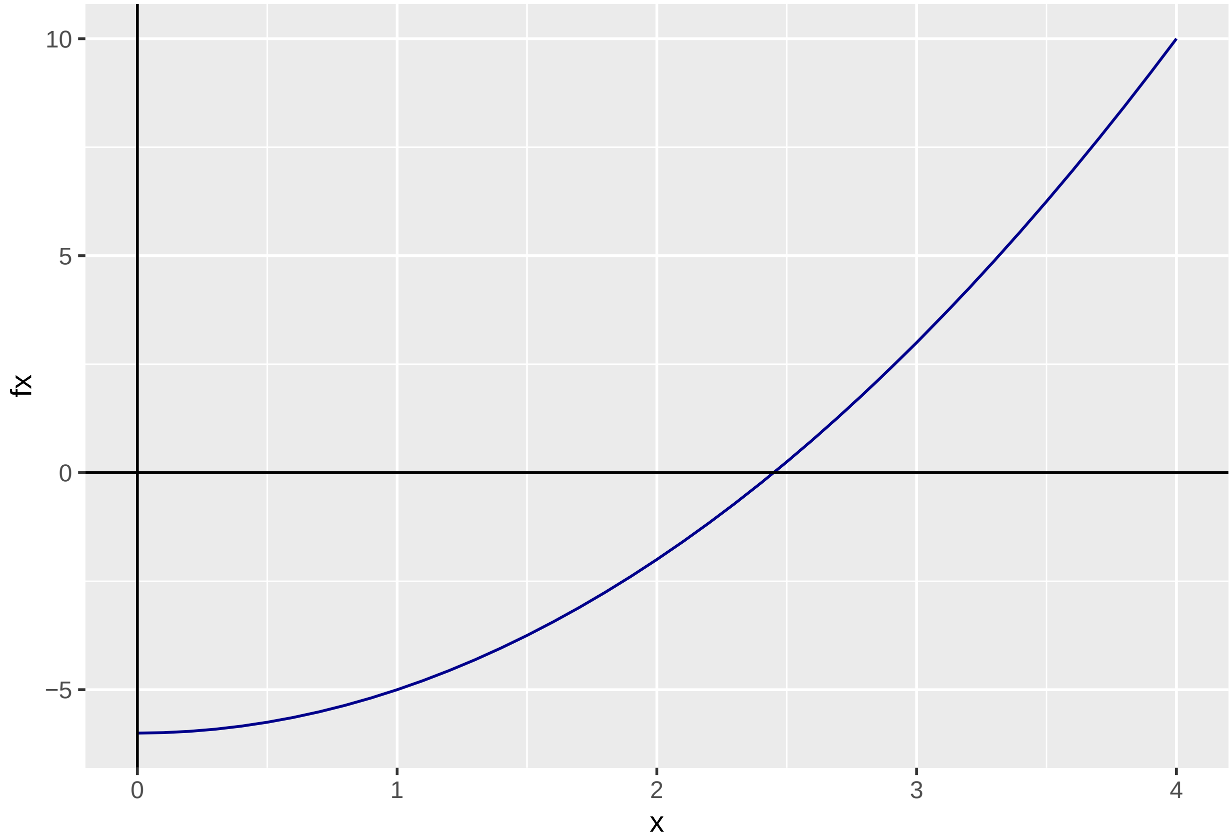
Ejercicio 7:

Sea $f(x) = x^2 - 6$. Con $p_0 = 3$ y $p_1 = 2$ encuentre p_3.

a) Aplique la fórmula de iteración de la secante.

```r
f <- function(x){
  return(x^2 - 6)
}

#Instancio un vector que me va a indicar los puntos en la función
x <- seq(0, 4, by = 0.1)

#Genero los puntos
fx <- f(x)

#Creo un data frame con los x e y
df <- data.frame(x, fx)

#Instancio los datos
gg_fx <- ggplot(data = df)

#Agrego la capa con los datos
gg_fx <- gg_fx + aes(x = x, y = fx)

#Est grafica una linea
gg_fx <- gg_fx + geom_line(linetype = 1, colour = "darkblue")

#Agrego el eje X
gg_fx <- gg_fx + geom_vline(xintercept = 0, linetype = 1)

#Agrego el eje Y
gg_fx <- gg_fx + geom_hline(yintercept = 0, linetype = 1)
```

```r
#Grafico
gg_fx
```




```r
Secante(p0 = 3, p1 = 2, tol = 0.1)
```

```
## [1] 2.454545
```

```r
FalsaPosicion(p0 = 3, p1 = 2, tol = 0.1)
```

```
## [1] 2.444444
```

Ejercicio 8:

Aplique el método de Newton para obtener soluciones con una exactitud de 0.0001 para los siguientes problemas:

a) $x^3 - 2 * x^2 - 5 = 0 \qquad [1; 4]$

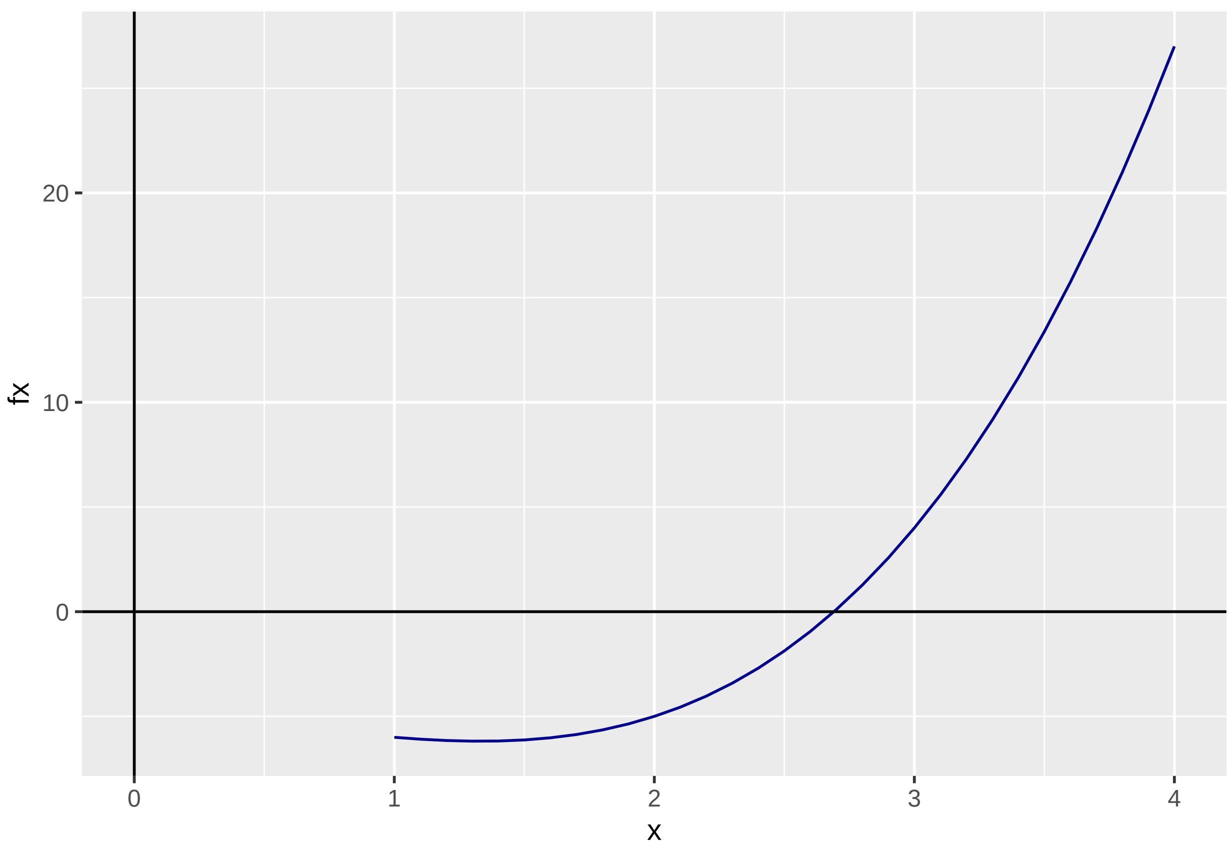
b) $x^3 - 3 * x^2 - 1 = 0 \qquad [-3; -2]$

c) $x - cos(x) = 0 \qquad [0; \frac{\pi}{2}]$

d) $x - 0.8 - 0.2sen(x) = 0 \qquad [0; \frac{\pi}{2}]$

```r
f <- function(x){
  return(x^3 - 2*x^2 - 5)
}
```

```r
#Instancio un vector que me va a indicar los puntos en la función
x <- seq(1, 4, by = 0.1)

#Genero los puntos
fx <- f(x)

#Creo un data frame con los x e y
df <- data.frame(x, fx)

#Instancio los datos
gg_fx <- ggplot(data = df)

#Agrego la capa con los datos
gg_fx <- gg_fx + aes(x = x, y = fx)

#Est grafica una linea
gg_fx <- gg_fx + geom_line(linetype = 1, colour = "darkblue")

#Agrego el eje X
gg_fx <- gg_fx + geom_vline(xintercept = 0, linetype = 1)

#Agrego el eje Y
gg_fx <- gg_fx + geom_hline(yintercept = 0, linetype = 1)

#Grafico
gg_fx
```



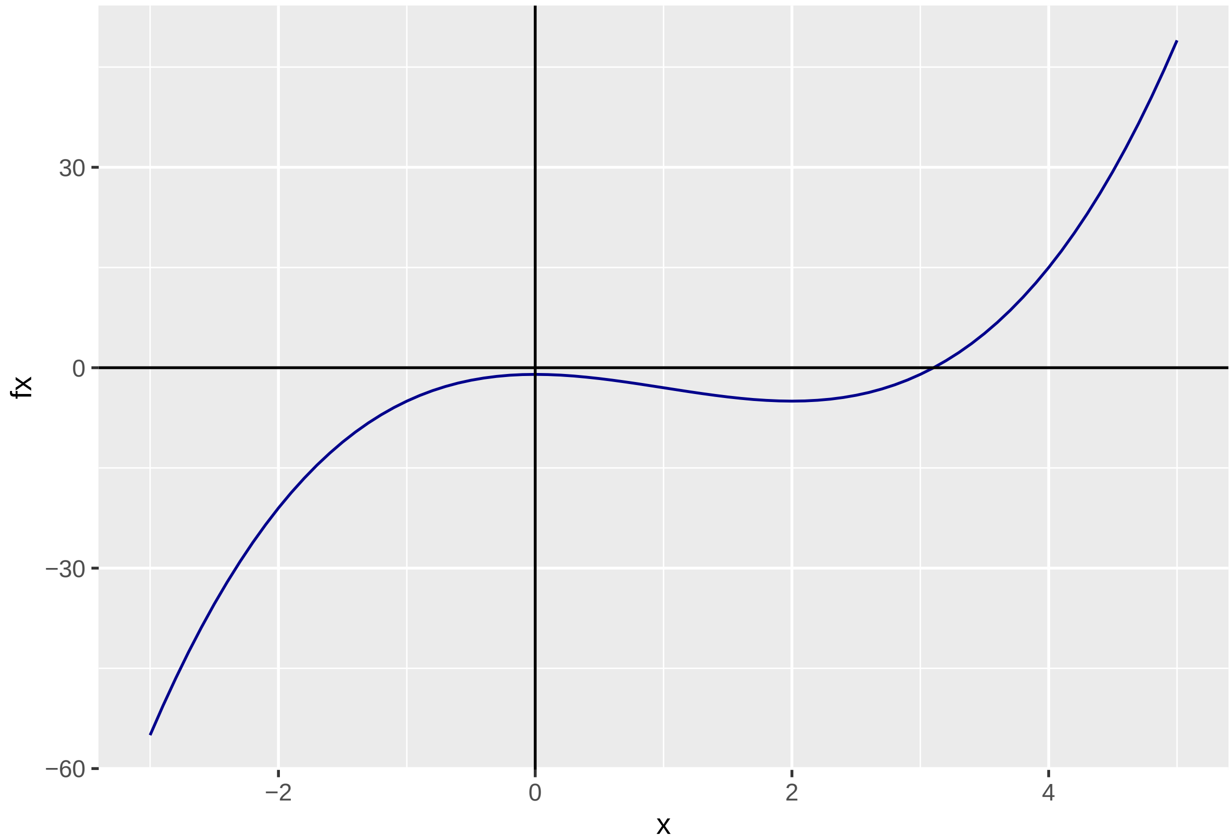
```r
fprima <- D(expression(x^3 - 2*x^2 - 5), "x")

fprima
```

```
## 3 * x^2 - 2 * (2 * x)
```

```r
fprima <- function(x){
  return(3 * x^2 - 2 * (2 * x))
}

Newton(p0 = 2.5, tol = 0.001)
```

```
## [1] 2.690647
```

```r
f <- function(x){
  return(x^3 - 3*x^2 - 1)
}

#Instancio un vector que me va a indicar los puntos en la función
x <- seq(-3, 5, by = 0.1)

#Genero los puntos
fx <- f(x)

#Creo un data frame con los x e y
df <- data.frame(x, fx)

#Instancio los datos
gg_fx <- ggplot(data = df)

#Agrego la capa con los datos
gg_fx <- gg_fx + aes(x = x, y = fx)

#Est grafica una linea
gg_fx <- gg_fx + geom_line(linetype = 1, colour = "darkblue")

#Agrego el eje X
gg_fx <- gg_fx + geom_vline(xintercept = 0, linetype = 1)

#Agrego el eje Y
gg_fx <- gg_fx + geom_hline(yintercept = 0, linetype = 1)

#Grafico
gg_fx
```




```
fprima <- D(expression(x^3 - 3*x^2 - 1), "x")

fprima
```

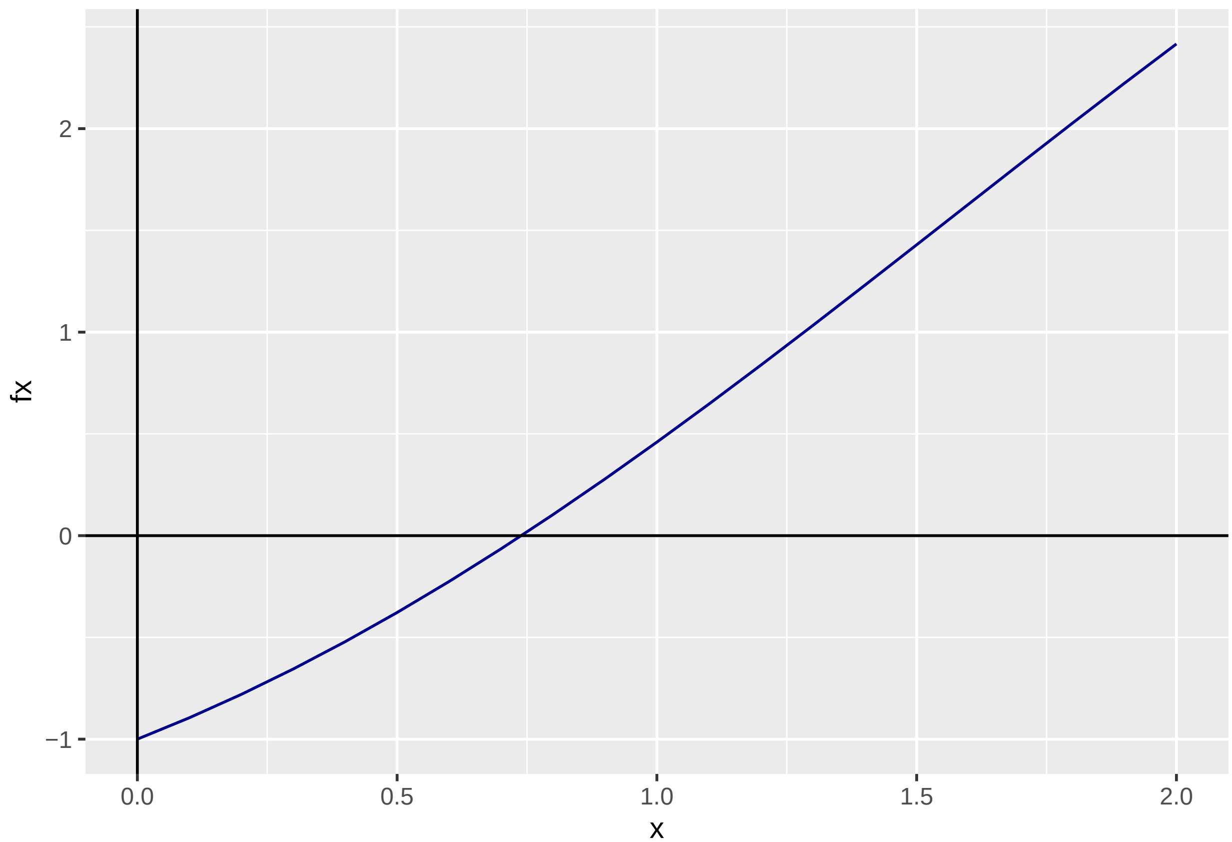
```
## 3 * x^2 - 3 * (2 * x)
```

```
fprima <- function(x){
  return(3 * x^2 - 3 * (2 * x))
}

Newton(p0 = 2.5, tol = 0.001)
```

```
## [1] 3.103803
```

```
f <- function(x){
  return(x - cos(x))
}

#Instancio un vector que me va a indicar los puntos en la función
x <- seq(0, 2, by = 0.1)

#Genero los puntos
fx <- f(x)

#Creo un data frame con los x e y
df <- data.frame(x, fx)
```

```r
#Instancio los datos
gg_fx <- ggplot(data = df)

#Agrego la capa con los datos
gg_fx <- gg_fx + aes(x = x, y = fx)

#Est grafica una linea
gg_fx <- gg_fx + geom_line(linetype = 1, colour = "darkblue")

#Agrego el eje X
gg_fx <- gg_fx + geom_vline(xintercept = 0, linetype = 1)

#Agrego el eje Y
gg_fx <- gg_fx + geom_hline(yintercept = 0, linetype = 1)

#Grafico
gg_fx
```




```r
fprima <- D(expression(x - cos(x)), "x")

fprima
```

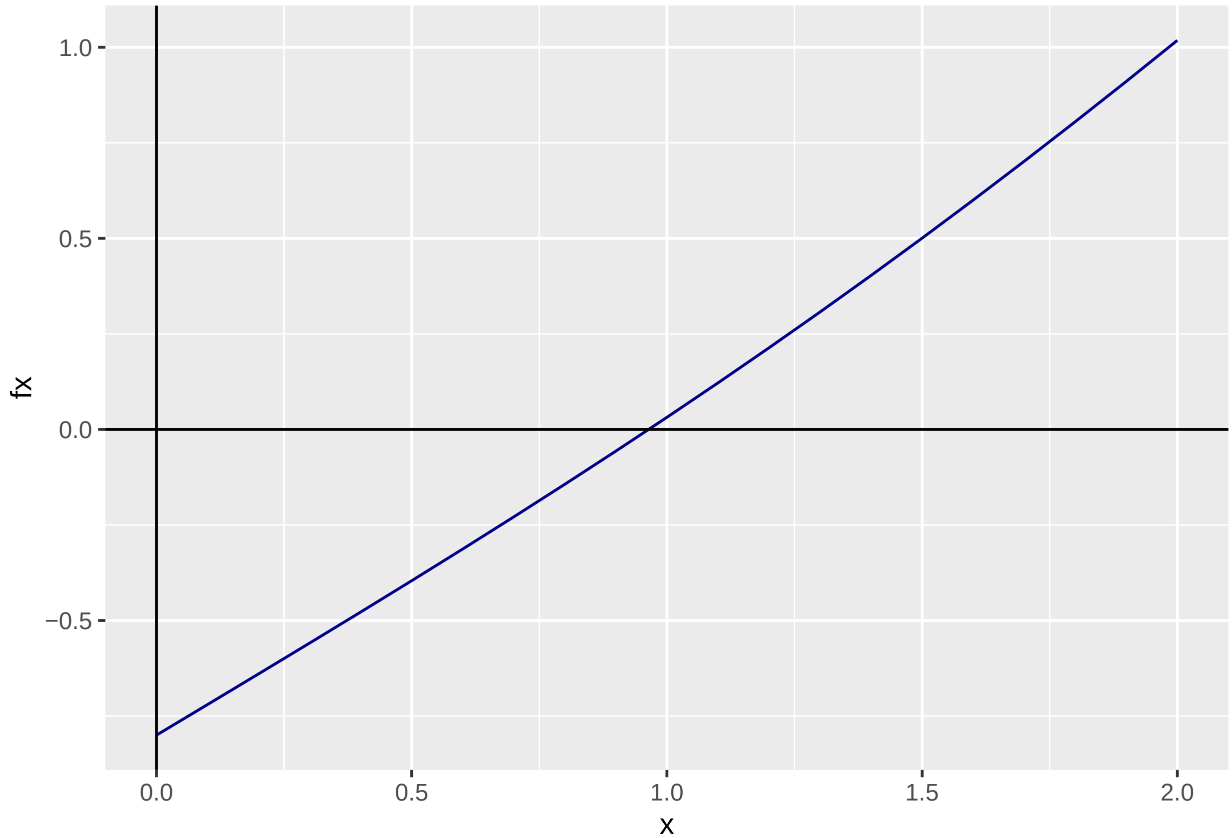
```
## 1 + sin(x)
```

```r
fprima <- function(x){
  return(1 + sin(x))
}
```

```r
Newton(p0 = 0.75, tol = 0.001)
```

```
## [1] 0.7390851
```

```r
f <- function(x){
  return(x - 0.8 - 0.2 * sin(x))
}

#Instancio un vector que me va a indicar los puntos en la función
x <- seq(0, 2, by = 0.1)

#Genero los puntos
fx <- f(x)

#Creo un data frame con los x e y
df <- data.frame(x, fx)

#Instancio los datos
gg_fx <- ggplot(data = df)

#Agrego la capa con los datos
gg_fx <- gg_fx + aes(x = x, y = fx)

#Est grafica una linea
gg_fx <- gg_fx + geom_line(linetype = 1, colour = "darkblue")

#Agrego el eje X
gg_fx <- gg_fx + geom_vline(xintercept = 0, linetype = 1)

#Agrego el eje Y
gg_fx <- gg_fx + geom_hline(yintercept = 0, linetype = 1)

#Grafico
gg_fx
```




```
fprima <- D(expression(x - 0.8 - 0.2 * sin(x)), "x")

fprima
```

```
## 1 - 0.2 * cos(x)
```

```
fprima <- function(x){
  return(1 - 0.2 * cos(x))
}

Newton(p0 = 0.75, tol = 0.001)
```

```
## [1] 0.9643339
```

Ejercicio 9:

Repita el ejercicio 8 usando (i) el método de la secante y (ii) el método de la posición falsa.Modifique las aproximaciones iniciales de ser necesario.

a)

```
Secante(p0 = 2.5, p1 = 3, tol = 0.001)
```

```
## [1] 0.9643359
```

```
FalsaPosicion(p0 = 2.5, p1 = 3, tol = 0.00001)
```

```
## [1] 0.9643353
```

b)

```r
Secante(p0 = 3, p1 = 4, tol = 0.0001)
```

```
## [1] 0.9643339
```

```r
FalsaPosicion(p0 = 3, p1 = 4, tol = 0.0001)
```

```
## [1] 0.9643384
```

c)

```r
Secante(p0 = 0.5, p1 = 0.75, tol = 0.0001)
```

```
## [1] 0.9643339
```

```r
FalsaPosicion(p0 = 0.5, p1 = 0.75, tol = 0.0001)
```

```
## [1] 0.9643339
```

d)

```r
Secante(p0 = 0.75, p1 = 1, tol = 0.0001)
```

```
## [1] 0.9643339
```

```r
FalsaPosicion(p0 = 0.75, p1 = 1, tol = 0.0001)
```

```
## [1] 0.9643339
```