Programación Estadística: Newton-Raphson

Adrián Sosa
Universidad Veracruzana

Newton-Raphson

El método de Newton-Raphson es un método abierto no está garantizada su convergencia global. La única manera de alcanzar la convergencia es seleccionar un valor inicial lo suficientemente cercano a la raíz buscada. La relativa cercanía del punto inicial a la raíz depende de la naturaleza de la propia función, si ésta presenta múltiples puntos de inflexión o pendientes grandes en el entorno de la raíz, entonces las probabilidades de que el algoritmo diverja aumentan.

El método linealiza la función por la recta tangente en ese valor supuesto. La abscisa en el origen de dicha recta será, según el método, una mejor aproximación de la raíz que el valor anterior. Se realizarán sucesivas iteraciones hasta que el método haya convergido lo suficiente.

Procedimiento

Para hallar una solución aproximada de $f(x) = 0$, dada una aproximación inicial p_0.

Entrada: aproximación inicial p_0 ; tolerancia TOL; cantidad máxima de iteraciones N; Salida: solución aproximada p ó mensaje de fracaso.

- Inicializar un contador $i = 1$;
- Mientras que $i \leq N$;
- Tomar $p = p_0 - \frac{f(p_0)}{f'(p_0)}$ Calculamos p .
- Si $|p - p0| < TOL$ entonces devolvemos p y terminamos;
- $i = i + 1$ incrementamos el contador.
- $p_0 = p$ redefinir p_0 .
- SALIDA('El método fracasó después de N iteraciones'); PARAR

Codificación

El método deNewton-Raphson es relativamente sencillo de implementar:

```
## function (Fx, df, p_0, maxiter = 50, tol = 1e-06)
## {
##     i <- 1
##     while (i <= maxiter) {
##         p <- p_0 - (Fx(p_0)/df(p_0))
##         if (abs(p - p_0) < tol) {
##             return(p)
##             break
##         }
##         p_0 <- p
##         i <- i + 1
##     }
## }
```

Para utilizarlo debemos definir una función para encontrar su ráiz y un espacio de búsqueda:

```
# función
fx1
```

```
## function (x)
## {
##     return(x^2 + 3 * x - 2)
## }
```

```
# derivada de la función
dfx1
```

```
## function (x)
## {
##     return(2 * x + 3)
## }
```

```
# se definen los límites
limS <- 10
limI <- -10
p_0 <- 5
steps <- 0.01
# se crea el espacio de búsqueda
x <- seq(limI, limS, steps)
y <- c()
for (i in seq_along(x)){y[i] <- fx1(x[i])}
# se optiene la ráiz
raiz <- newton(fx1,dfx1, p_0)
```

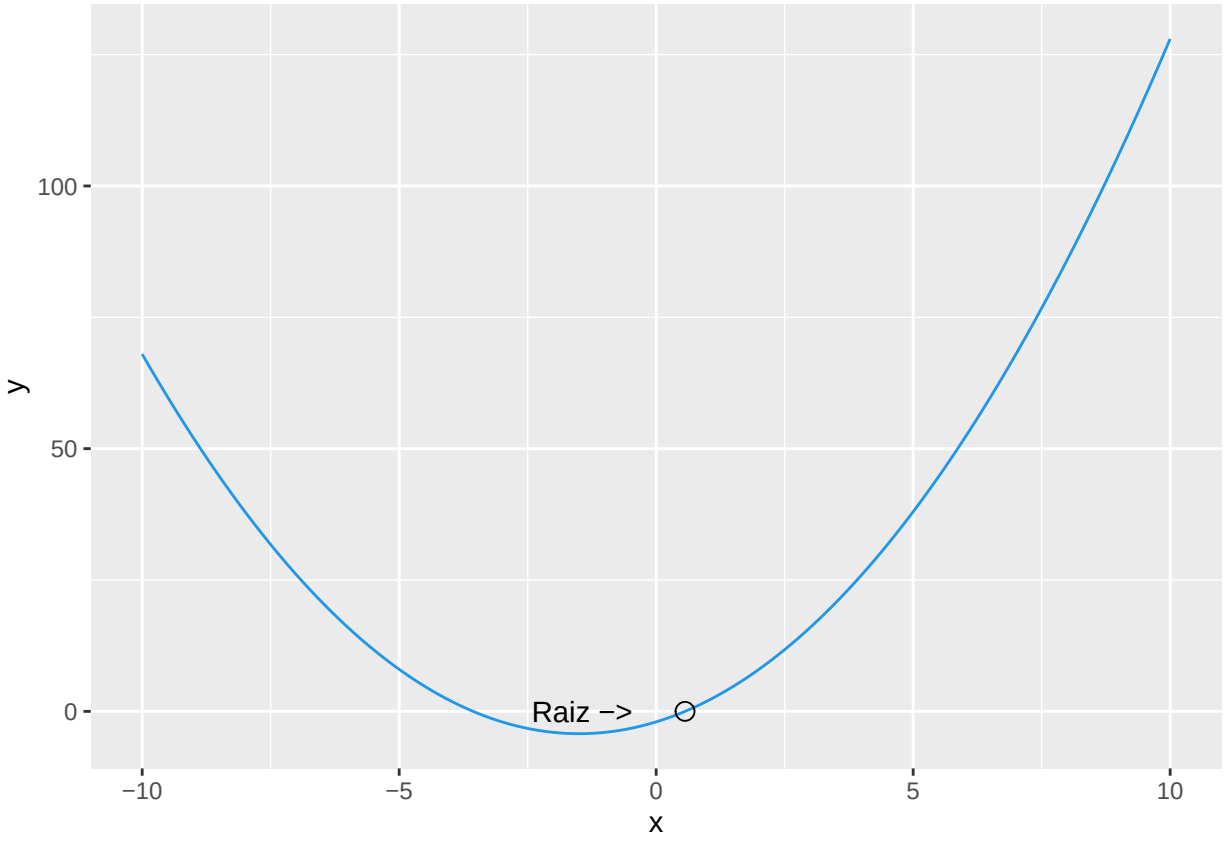
El resultado se almacenara en la variable *raiz,* para conocerlo basta con imprimirlo:

```
raiz
```

```
## [1] 0.5615528
```

Si deseamos podémos gráficar la función y nuestro resultado.

```
# se crea un dataframe
df <- data.frame(x,y)
ggplot(data= df, mapping=aes(x= x, y =y))+
  geom_line(color=4)+
  annotate(geom="text",x=raiz-2, y=0, label="Raiz ->")+
  annotate(geom="point", x=raiz,y =0, size =3, shape=1, fill="transparent")
```



Si deseamos podémos calcular y gráficar ambas raices de la función.

```
raiz_superior <- newton(fx1,dfx1, limS)
raiz_inferior <- newton(fx1,dfx1, limI)
ggplot(data= df, mapping=aes(x= x, y =y))+
  geom_line(color=4)+
  annotate(geom="text",x=raiz_superior+2, y=0, label=" <- Raiz superior ")+
  annotate(geom="point", x=raiz_superior,y =0, size =3, shape=1, fill="transparent")+
  annotate(geom="text",x=raiz_inferior-2, y=0, label=" Raiz inferior -> ")+
  annotate(geom="point", x=raiz_inferior,y =0, size =3, shape=1, fill="transparent")
```