Integración Numérica Aplicada a Modelos de Sobrevivencia de Makeham.

Autores:
García Melena Ethan Leonel
Guerrero Suárez Luis Miguel

Motivación

Los modelos de mortalidad de Makeham son comunes para el calculo de funciones de sobrevivencia en el ambito actuarial especificamente en seguros, este modelo, parte de la idea que las personas morimos por dos causas, una es un factor al azar, es decir, una personas puede morir sin necesidad de estar enfermo, su muerte, es debido a un accidente, y el otro factor es la resistencia de las personas a la muerte, el cual con el paso de los años va decreciendo. Este modelo se usa principalmente para el calculo de funciones de sobrevivencia de personas adultas ya que en edades muy tempranas mortalidad es decreciente. Al ser un modelo complejo, el calcular una función de sobrevivencia es casi imposible, ya que no existe antiderivada para tales calculos, entonces, optamos por una aproximacón. El porque escogimos este tema, es debido a que en algún momento nos topamos con un seguro que suponía un modelo de makeham el cual no pudimos resolver.

Planteamiento y Propuesta de solución al Problema:

Como se mencionó anteriormente hay un cierto nivel de dificultad para poder calcular las probabilidades de sobrevivencia cuando se tienen modelos de Makeham pues las integrales para poder calcular las ${}_tP_x$ se pueden tornar un poco complicadas. Es por eso que la intención en este proyecto es poder resolver estas integrales numéricamente bajo los métodos de:

- Trapecio simple y compuesto.
- Simpson 1/3 simple y compuesto.
- Simpson 3/8 simple y compuesto.
- Cuadratura Gaussiana.

De tal forma que veamos cuál es la mejor aproximación y usar ese método para la resolución de estas integrales y así obtener de una manera más rápida y sencilla estas probabilidades de sobrevivencia.

Hipótesis.

Ejecutaremos los algoritmos de: trapecio simple, trapecio compuesto, Simpson 1/3 simple, Simpson 1/3 compuesto, Simpson 3/8 simple y Simpson 3/8 compuesto, para la resolución de nuestra problemática, nuestra hipótesis es que el método de Cuadratura Gaussiana será el que más se aproxime a a nuestr resultado y que al usar un metodo que contemple más nodos o particiones, mejor será nuestra aproximación.

Creamos la función.

```
f1 <- function(x){
  .0001+.0003*(1.075)^(20+x)
}
```

Método de Trapecio Simple

```
# Método del trapecio simple
# Suponemos que solo conocemos 2 nodos
trapecio_simple <- function(f,x0,x1){
  q <- (x1-x0)/2 * (f(x1) + f(x0))
  x_i <- c(x0,x1)
  f_x_i <- c(f(x0),f(x1))
  i_s <- c(0,1)
  Tabla <- data.frame(i_s,f_x_i,x_i)
  colnames(Tabla) <- c("i","xi","f(xi)")
  print(Tabla)
  cat("\n","La aproximación numérica de la integral es:",q)
}
trapecio_simple(f1,0,10)
```

```
##   i          xi f(xi)
## 1 0 0.001374355     0
## 2 1 0.002726487    10
##
##  La aproximación numérica de la integral es: 0.02050421
```

Método de Trapecio Compuesto

```
trapecio_compuesto <- function(f, a, b, n) {
  if (is.function(f) == FALSE) {
    cat("f debe ser una función.")
    break
  }
  h <- (b - a) / n
  x_i <- seq.int(a, b, by = h)
  f_x_i <- c(f(x_i))
  i_s <- seq.int(0,n,by=1)
  u <- c()
  for(i in 1:(n-1)){
    u <- c(u,f(x_i[i+1]))
  }
  SumaU <- sum(u)
  aproximacion <- (h/2)*(f(a)+(2*SumaU)+f(b))
  Tabla <- data.frame(i_s,x_i,f_x_i)
  colnames(Tabla) <- c("i","xi","f(xi)")
  print(Tabla)
  cat("\n","La aproximación numérica de la integral es:",aproximacion)
}
trapecio_compuesto(f1, 0, 10, 3)
```

```
##   i        xi       f(xi)
## 1 0  0.000000 0.001374355
## 2 1  3.333333 0.001721756
## 3 2  6.666667 0.002163860
## 4 3 10.000000 0.002726487
##
##  La aproximación numérica de la integral es: 0.01978679
```

Método de Simpson 1/3 Simple

```
# Suponemos que solo conocemos 3 nodos
Simpson_Simple_1_3 <- function(f,x0,x1){
  h <- abs(x1-x0)/2
  x_i <- c(x0,(x0+h),x1)
  f_x_i <- c(f(x0),f(x0+h),f(x1))
  i_s <- c(0,1,2)
  q <- (h/3) * (f(x0) + 4* f(x0 + h) + f(x1))
  Tabla <- data.frame(i_s,x_i,f_x_i)
  colnames(Tabla) <- c("i","xi","f(xi)")
  print(Tabla)
  cat("\n","La aproximación numérica de la integral es:",q)
}
Simpson_Simple_1_3(f1,0,10)
```

```
##   i xi       f(xi)
## 1 0  0 0.001374355
## 2 1  5 0.001929502
## 3 2 10 0.002726487
##
##  La aproximación numérica de la integral es: 0.01969808
```

Método de Simpson 1/3 Compuesto

```
Simpson_Compuesto_1_3 <- function(f, a, b, n){
  if (is.function(f) == FALSE) {
    print('f debe ser una función, por favor ingresar una función f sin evaluar.')
    break
  }
  h <- (b - a) / n
  x_i <- seq.int(a, b, by = h)
  f_x_i <- c(f(x_i))
  i_s <- seq.int(0,n,by=1)
  par <- c()
  impar <- c()
  for(i in 1:(n-1)){
    if(i%%2==0){
      par <- c(par,f(x_i[i+1]))
    }else if(i%%2!= 0){
      impar <- c(impar,f(x_i[i+1]))
    }
  }
  Sumapar <- sum(par)
  Sumaimpar <- sum(impar)
  aproximacion <- (h/3)*(f(a)+(4*Sumaimpar)+(2*Sumapar)+f(b))
  Tabla <- data.frame(i_s,x_i,f_x_i)
  colnames(Tabla) <- c("i","xi","f(xi)")
  print(Tabla)
  cat("\n","La aproximación numérica de la integral es:",aproximacion)
}
Simpson_Compuesto_1_3(f1, 0, 10, 4)
```

```
##   i   xi       f(xi)
```

```
## 1 0  0.0 0.001374355
## 2 1  2.5 0.001626904
## 3 2  5.0 0.001929502
## 4 3  7.5 0.002292068
## 5 4 10.0 0.002726487
##
##  La aproximación numérica de la integral es: 0.01969644
```

Método de Simpson 3/8 Simple

```
# Suponemos que conocemos 4 nodos
Simpson_Simple_3_8 <- function(f,x0,x1){
  h <- abs(x1-x0)/3
  x_i <- c(x0,(x0+h),(x0+2*h),x1)
  f_x_i <- c(f(x0),f(x0+h),f(x0+2*h),f(x1))
  i_s <- c(0,1,2,3)
  q <- 3*h/8 * (f(x0) + 3 * f(x0+h) + 3 * f(x0 + 2 * h) + f(x1))
  Tabla <- data.frame(i_s,x_i,f_x_i)
  colnames(Tabla) <- c("i","xi","f(xi)")
  print(Tabla)
  cat("\n","La aproximación numérica de la integral es:",q)
}
Simpson_Simple_3_8(f1,0,10)
```

```
##   i        xi       f(xi)
## 1 0  0.000000 0.001374355
## 2 1  3.333333 0.001721756
## 3 2  6.666667 0.002163860
## 4 3 10.000000 0.002726487
##
##  La aproximación numérica de la integral es: 0.01969711
```

Método de Simpson 3/8 Compuesto

```
# Suponemos que se ingresará un numero
# de nodos tales que el número de intervalos es multiplo de 3
Simpson_Compuesto_3_8 <- function(f,n,x0,xn){
  #construimos h
  z <- abs(xn-x0)/(n -1)
  #iniciamos un vector vacío
  e <- c()
  #Calculamos las imagenes de cada nodo y las guardamos en un vector
  for (i in 0:(n-1)){
    q <-f(x0+i*z)
    e <- c(e,q)
  }
  #Hacemos la suma de de f(x3i)
  r <- 0
  for(i in 1:(((n-1)/3)-1)){
    t <- e[3*i+1]
    r <- r+t
  }
  #hacemos la suma de f(x3i+1) + f(x3i+2)
  u <- 0
```

```
  for(i in 0:(((n-1)/3)-1)){
    y <- e[(3*i+2)] + e[(3*i+3)]
    u <- u+y
  }
  #Finalmente ejecutamos la formula y obtenemos el resultado
  x <- (3*z/8)*(e[1] + (2 * r) + (3 * u)+e[n])
  x_i <- seq.int(x0,xn,by = z)
  f_x_i <- e
  i_s <- seq.int(0,n-1, by=1)
  Tabla <- data.frame(i_s,x_i,f_x_i)
  colnames(Tabla) <- c("i","xi","f(xi)")
  print(Tabla)
  cat("\n",
      "La aproximación numérica de la integral es:",x)
}
Simpson_Compuesto_3_8(f1,7,0,10)
```

```
##   i        xi       f(xi)
## 1 0  0.000000 0.001374355
## 2 1  1.666667 0.001537600
## 3 2  3.333333 0.001721756
## 4 3  5.000000 0.001929502
## 5 4  6.666667 0.002163860
## 6 5  8.333333 0.002428240
## 7 6 10.000000 0.002726487
##
##  La aproximación numérica de la integral es: 0.01969638
```

Método de Cuadratura Gaussiana.

```
Cuadratura_Gaussiana <- function(f,a,b,n){
  #construimos una matriz con los wi's correspondientes
  m1 <- matrix(c(1,1,0,0,0,.5555555556,.8888888889,.5555555556,0,0,.3478548451,
  .6521451549,.6521451549,.3478548451,0,.236926885,.4786286705,.5688888889,
  .4786286705,.236926885),5,4)
  #Construimos una matriz con los xi's correspondientes
  m2 <- matrix(c(-sqrt(1/3),sqrt(1/3),0,0,0,.7745966692,0,-.7745966692,0,0,
.8611363116,.3399810436,-.3399810436,-.8611363116,0,.9061798459,.5384693101,0,
-.5384693101,-.9061798459),5,4)
  #creamos vectores vacos para poderlos llenar de acuerdo al numero de nodos
  #wi's
  w <- rep(0,n)
  #f(xi)'s
  x <- rep(0,n)
  #Caso cuando nuestra integral va de -1 a 1
  if(a == -1 && b == 1){
    # Llenamos el vector w con sus valores correspondientes de la matriz m1
    # dependiendo el valor de n Llenamos el vector x con los valores de f(x)
    # evaluadas en los x de la matrix m2
    for(i in 1:n){
      w[i] <- m1[i,n-1]
      x[i] <- f(m2[i,n-1])
    }
```

```r
    #multiplicamos entrada por entrada de los vectores x y w y sumamos.
    s <- sum(w*x)
    cat("La aproximación es", s)
  }else{
    #En caso que a != -1 ó b != 1
    # Llenamos el vector w con sus valores correspondientes de
    # la matriz m1 dependiendo el valor de n
    # Llenamos el vector x con los valores de f() evaluadas en la tranformación
    for(i in 1:n){
      w[i] <- m1[i,n-1]
      x[i] <- f((((b-a)*m2[i,n-1])+a+b)/2)
    }
    #multiplicamos entrada por entrada de los vectores x y w y sumamos a
    # esto lo multuplicamos por el
    #tamaño del intervalo dividido entre el 2 del diferencial
    s <- ((b-a)/2)*sum(w*x)
    cat("La aproximación es", s)
  }
}
Cuadratura_Gaussiana(f1,0,10,5)
```

```
## La aproximación es 0.01969633
```

Analisis de Resultados.

Ley empleada	Resultado	Resultado Real	Error absoluto	Error relativo
Trapecio simple	0.02050421	0.0196963	0.00080791000	-0.041018364
Trapecio compuesto	0.01978679	0.0196963	0.00009049000	-0.004594264
Simpson 1/3 simple	0.01969808	0.0196963	0.00000178000	-9.03723E-05
Simpson 1/3 compuesto	0.01969644	0.0196963	0.00000014000	-7.10793E-06
Simpson 3/8 simple	0.01969711	0.0196963	0.00000081000	-4.11245E-05
Simpson 3/8 compuesto	0.01969638	0.0196963	0.00000008000	-4.06168E-06
Cuadratura Gaussiana	0.01969633	0.0196963	0.00000003000	-1.52313E-06

Figure 1: Resultados de las Aproximaciones.

Conclusiones.

Como podemos observar en la tabla anterior los métodos que mejor se aproximan son Simpson 3/8 compuesto y la Cuadratura Gaussiana ya que el error es mínimo si lo comparamos con el resultado real almenos los primeros 7 decimales son iguales, esto es dibido a que en este caso supusimos 7 nodos, pero si aumentamos la cantidad de nodos será aún más preciso de lo que ya es usando 7. Este método es más facil de emplear que tratar de resolver la integral. En todos los métodos al emplearlos de manera compuesta son más precisos que si los hacemos de manera simple y esto es debido a que en el compuesto se pueden emplear muchos más nodos que el simple y con esto se generan más particiones de nuestra región lo cual genera una mejor aproximación.