

## 統計模擬 作業四

109354003 統碩一 吳書恆

109354027 統碩一 蔡海蓮

5/18/2021

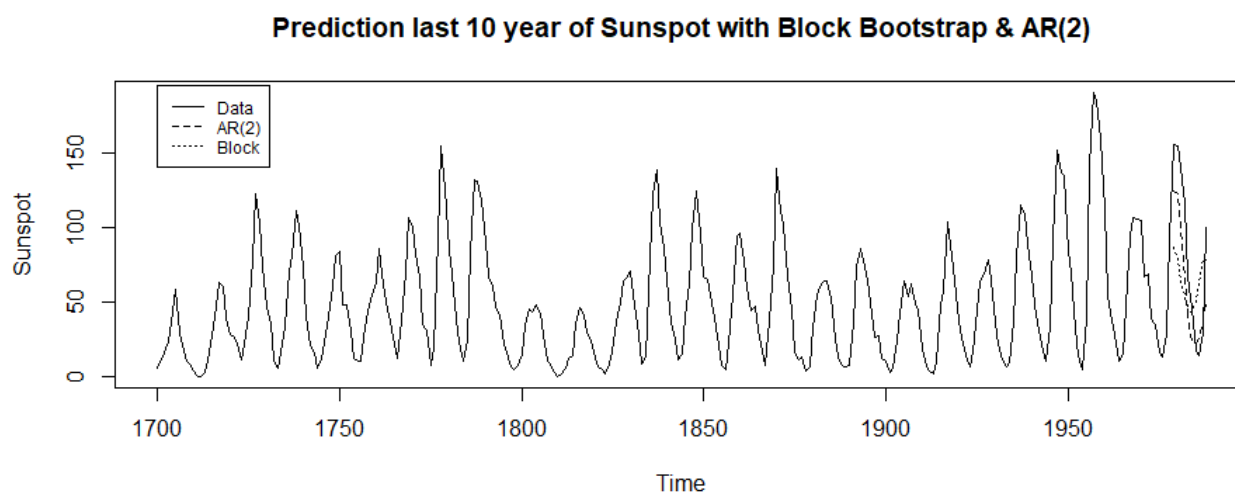
1. The block bootstrap can be used in prediction for dependent data. Use the built-in data “sunspot.year” in R, which is can be modeled as an AR(2) model, compare the difference of prediction via block bootstrap and AR(2) model. As a check, you can leave the final 10 observations as “testing” data.

首先將資料拆成 training 和 testing 的部分，計算 training 的 block bootstrap 和 AR(2)，再分別預測 10 年的變化，並比較與 testing 的差異。

block bootstrap: 先計算相鄰時間觀察值的差值，之後抽出一整個區塊差值，計算中位數後，加到最後一期的觀察值。以下指令示範如何抽取不重疊區集，並取中位數。

```
for(i in 1:10000){  
  sa <- sample(1:27, 1)*10-9  
  zeroma[i,] <- dr[c(sa:(sa+9))]  
}  
md <- apply(zeroma, 2, median)
```

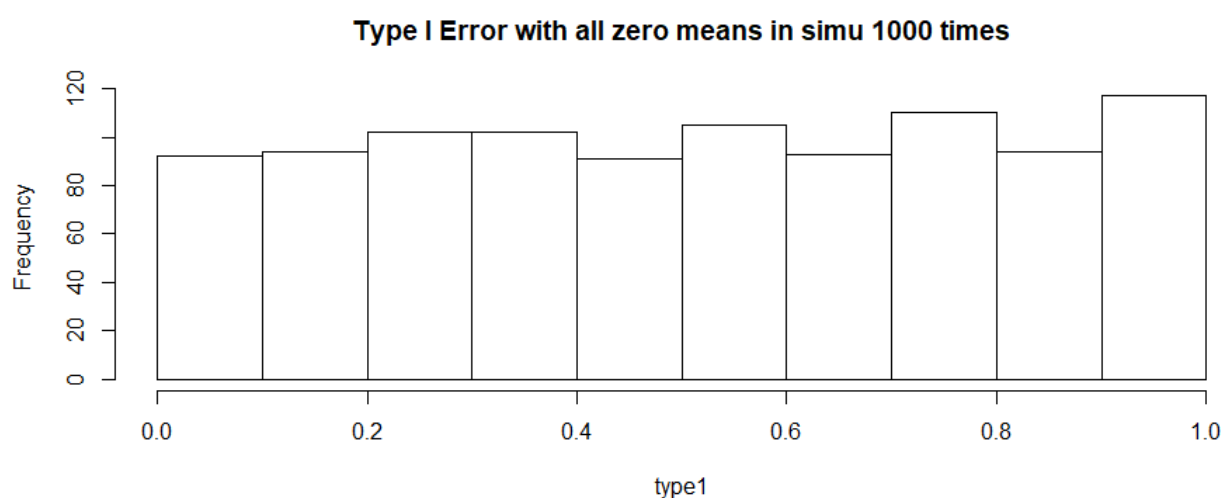
下圖為依據 10000 次區塊拔靴法的模擬，與差分後的 AR(2)模型之預測值之比較。(請看下圖最右邊的變動)



實線是真實資料，可以比較出，AR(2)預測結果較符合實際資料。

2. Similar to what we saw in class, use simulation to evaluate the type I error and testing powers of one-way ANOVA. Suppose there are 3 treatments, each with 16 observations. However, assume that the treatment with larger mean has variance 2, instead of 1. (Note: This assignment is to check the influence of the constant variance assumption. Also, as an extended study, you could also check the normality and independence assumptions.)

在虛無假設為真的情況之下（三組平均數皆為 0），型一誤差即為 Anova 檢定的 p 值，因此先隨機產生三組  $N(0, 1)$  的亂數，並計算每次的 p 值，共循環 1000 次，得到結果如下。



1000 次模擬的 p-value 是均勻分配，ks 檢定的 p 值為 0.5225。說明型一誤差在模擬的母體均數相等、常態與變異數假設不違反的狀況之下，並不會有不穩定的情形發生。

接著計算 power，在對立假設為真的情況之下（這邊假設其中一組平均數為大於等於 0），檢定統計量服從 noncentral F-test( $\nu_1, \nu_2, \lambda$ )， $\nu_1, \nu_2$  分為  $MStr$  與  $MSE$  的自由度， $\lambda$  為非中心化參數，

$$\lambda = \frac{r \sum_{i=1}^g \alpha_i^2}{\sigma^2}$$

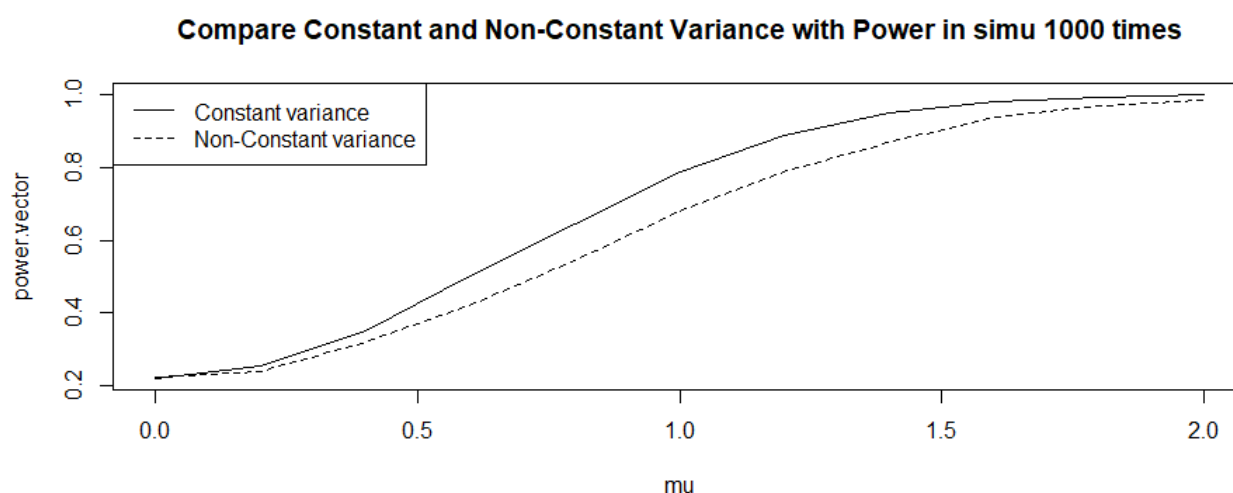
其中， $r$  為每組的樣本數， $\alpha_i$  為處理效果，可以推得  $SStr = r \sum_{i=1}^g \alpha_i^2$ ，則以下式子成立，

$$\lambda = \frac{SStr}{MSE} = \frac{(g-1)MStr}{MSE} = (g-1)F_0$$

有了以上關係，我們可以計算出在不同的對立假設情形之下的 power 變化，這邊需假設顯著水準固定的情況之下（這邊設 0.05），如以下指令：

```
power <- pf(qf(.95, 2, 45), 2, 45, ncp=F.value*2, lower.tail = F)
```

為了比較變異數假設有無違反的情形，我們將平均數較大組別的變異數從 1 放大為 2，比較結果如下圖（下一頁上方），發現在變異數違反的情況之下的 power 普遍都比不違反來的低一些。



3. Use the bisection, false positions, and/or secant methods to find the roots, and check your answers with the functions in R (e.g., “uniroot”). You need to specify the starting points, convergence criterion, and number of iterations.

(a)  $f(x) = x^3 + 2x^2 + 3x - 1$ ;

(b)  $f(x) = e - \frac{1}{3.5 + x}$ ;

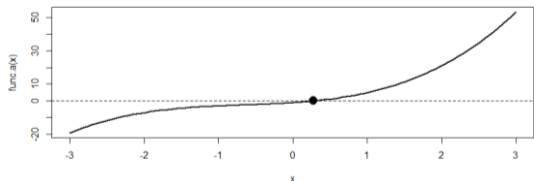
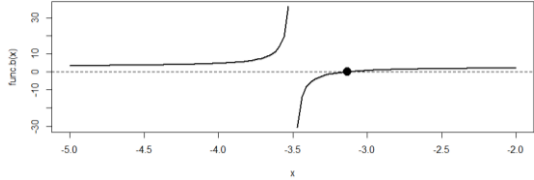
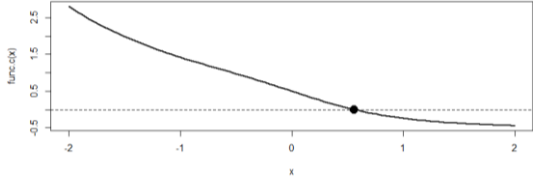
(c)  $f(x) = \frac{e^{-x}}{\sqrt{1+x^2}} - 0.5$ .

本題使用了 Bisection 和 False positions 方法來解以上函數的根，再用 R 裡頭的函數 uniroot 來驗證根的結果。Bisection 的概念在於切割與勘根定理，函數會列在文末；False positions 是用到通過任兩點  $(x_1, x_2)$  的直線與 X 軸的交點  $(x_3)$  來逼近根，以下式子成立：

$$x_3 = x_1 - f(x_1) \frac{x_1 - x_2}{f(x_1) - f(x_2)}$$

False positions 與 Scent 的差異在於有無固定任兩點的其中一點，這邊固定  $x_1$ ，在下一個迭代  $x_3$  將取代原本  $x_2$  的位置，從新計算出通過該兩點的直線與 X 軸的交點  $(x_4)$ ，……。以下整理了三個方法找出的根與圖形。以下收斂的標準為  $10^{-6}$ 。

以下有幾點要注意，首先，由於函數(b)在  $x = -3.5$  附近存在漸進線，因此對於初始值的給定在  $-3.5$  附近會影響根有無求出。

方法	初始值	根	迭代次數	圖形
函數(a)				
Bisection	(-4, 3)	0.2756823	21	
False posi-	(-4, 3)	0.2756811	30	
uniroot	(-4, 3)	0.2756814	9	
函數(b)				
Bisection	(-3.4, -2)	-3.132121	19	
False posi-	(-3.4, -2)	-3.132118	41	
uniroot	(-3.4, -2)	-3.132141	6	
函數(c)				
Bisection	(-1, 2)	0.5577289	20	
False posi-	(-1, 2)	0.5577288	12	
uniroot	(-1, 2)	0.5577287	7	

4. Consider a multinomial observation  $X = (x_1, x_2, x_3, x_4)$  with class probabilities given by  $(p_1, p_2, p_3, p_4) = (\frac{2+\theta}{4}, \frac{1-\theta}{4}, \frac{1-\theta}{4}, \frac{\theta}{4})$ , where  $0 < \theta < 1$ . The sample size is  $n = \sum x_i$  and the parameter  $\theta$  is to be estimated from the observed frequencies (1997, 906, 904, 32), i.e., sample size 3839. Use the secant, Ridder's (or Brent's), and Newton-Raphson methods to find the MLE (via  $l'(\theta)$ ). You may choose your own starting points and convergence criterion (preferred  $10^{-6}$  or smaller).

已知四個參數多項分布 (multinomial distribution) 的概似函數為

$$L(p_1, \dots, p_4 | x_1, \dots, x_4) = \binom{n}{x_1, \dots, x_4} \prod_{i=1}^4 p_i^{x_i}$$

且

$$l(p_1, \dots, p_4 | x_1, \dots, x_4) = \log n! - \sum_{i=1}^4 \log x_i! + \sum_{i=1}^4 x_i \log p_i$$

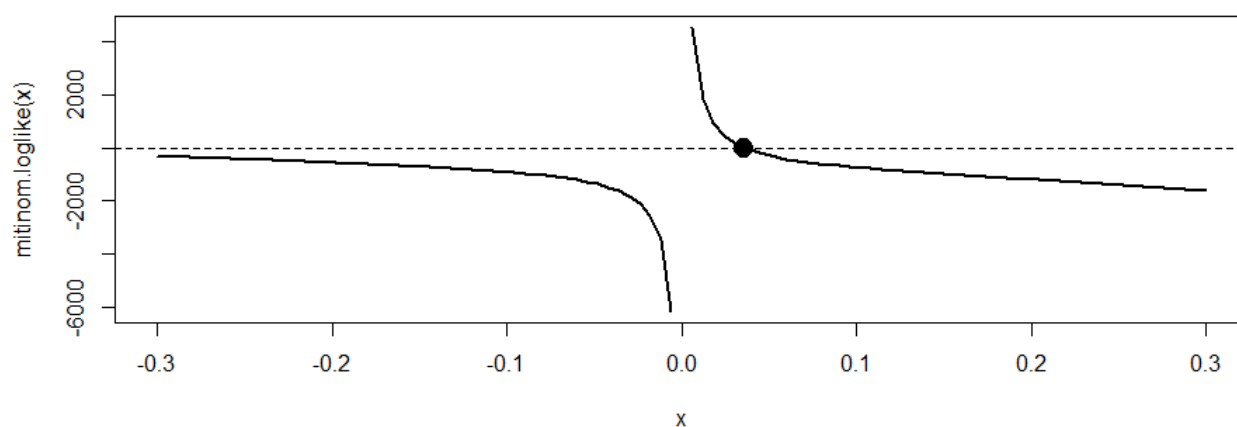
將  $(p_1, p_2, p_3, p_4)$  替換成  $(\frac{2+\theta}{4}, \frac{1-\theta}{4}, \frac{1-\theta}{4}, \frac{\theta}{4})$ , 得到

$$l(\theta | x_1, \dots, x_4) = (\text{constant to } \theta) + x_1 \log \left( \frac{2+\theta}{4} \right) + \dots + x_4 \log \left( \frac{\theta}{4} \right)$$

取其一階導函數並帶入  $(x_1, x_2, x_3, x_4) = (1997, 906, 904, 32)$  為

$$l(\theta | x_1, \dots, x_4) = \frac{1997}{2+\theta} - \frac{906}{1-\theta} - \frac{904}{1-\theta} + \frac{32}{\theta}$$

此題要求對數最大概似函數最大值，即求一階導函數為 0 的根 ( $\theta$  需介在 0 到 1 之間)，圖形如下。



透過自己編寫的 Secant、Ridder 和 Newton.Raphson 函數，皆可找到其根。

方法	初始值	根	迭代次數
Secant	(-.01, .99)	0.0357123	8
Ridder	(-.01, .99)	0.0357123	7
Newton.Raphson	(-.01, .99)	0.0357123	6

注意，這邊的初始值要避開漸進線才可以執行。

Secant 方法類似 False positions，這邊就不再多做說明。Ridder 則是參考維基百科給的定義下自行編寫函數<sup>1</sup>。Newton.Raphson 需要作微分，在 R 裏頭的套件叫做 numDeriv 需要用到。函數編寫如下：

```
newton.raphson <- function(f, init, maxiter=1000, tol=1e-06){
  require(numDeriv)
  if (f(init) == 0.0) {return(init)}
  for (i in 1:maxiter) {
    dx <- genD(func = f, x = init)$D[1]
    init.next <- init - f(init)/dx
    if (abs(init.next - init) < tol) break
    init <- init.next
  }
  list(root = init.next, f.root = f(init.next), iter = i)
}
```

<sup>1</sup> Ridders' method in Wikipedia: [https://en.wikipedia.org/wiki/Ridders%27\\_method](https://en.wikipedia.org/wiki/Ridders%27_method)

5. Try at least three different methods to find the estimates of  $B$  and  $C$  for the Gompertz model,  $\mu_x = BC^x$ ,  $x > 0$ , using the Taiwan data in 2018-2020. You may count “nlminb”, “nls” or “opt” as one of the methods (for replacing Newton’s method). Also, similar to what we saw in the class, discuss the influence of starting points to the number of iterations. You may choose the male data or female data. (Bonus: Compare the results of different counties.)

資料來自內政部 2017 ~ 2019, 共 3 年的死亡率資料, 並合併存活人口資料換算出死亡人數。若資料滿足 Gompertz 假設, 使用 WLS 找最小值須滿足以下式子,

$$\min_{\alpha, \beta} \sum_x w_x (\log(-\log p_x) - \alpha - \beta x)^2$$

其中  $\alpha = \log B + \log(C - 1) - \log(\log C)$  和  $\beta = \log C$ 。

透過 nlminb 可得到  $B = 0.00011$ ,  $C = 1.08137$ ,  $SSE = 15078.4$ ,  $MSE = 50.09435$ 。

至於另外兩個方式 (NM 和 MLE) 來估計  $B$  和  $C$ , 但我們無法找到適合的起始點來收斂, 可能性是因為 Gompertz 假設較適合用在高年齡存活率的估計, 而我們的資料年齡是包含所有年齡層, 可能在去除年輕的年齡層後會有較好的表現。

Code:

```

#HW4
#1#####
data <- sunspot.year
train.data <- data[9:279]
test.data <- data[280:289]

zeroma <- matrix(0, 10000, ncol = 10)
dr <- train.data[-1] - train.data[-271]

for(i in 1:10000){
  sa <- sample(1:27, 1)*10-9
  zeroma[i,] <- dr[c(sa:(sa+9))]
}
md <- apply(zeroma, 2, median)
predict.value <- NULL
for(i in 1:10){
  predict.value <- c(predict.value, data[279] + sum(md[c(1:i)]))
}
b <- ts(predict.value, frequency = 1 ,start = c(1979,1))

x <- ar.ols(train.data, order = 2)
z <- predict(x, n.ahead = 10)
a <- ts(z$pred, frequency = 1 ,start = c(1979,1))

ts.plot(data, a, b, ylab = "Sunspot", lty = 1:3,
  main = "Prediction last 10 year of Sunspot with Block Bootstrap
& AR(2)")
legend(1700, 195, lty = 1:3, legend = c("Data", "AR(2)", "Block"), cex
= 0.75)

#2#####
type1.test <- function(n){
  type1.vector <- NULL
  for(i in 1:n){
    x <- cbind(rep(1:3, each = 16), rnorm(48))
    fit <- lm(x[, 2] ~ factor(x[, 1]))
    type1 <- anova(fit)$"Pr(>F)"[1]
    type1.vector <- c(type1.vector, type1)
  }
  return(type1.vector)
}

type1 <- type1.test(n=1000)
hist(type1, main = "Type I Error with all zero means in simu 1000
times")
ks.test(type1, 'punif')

power.test <- function(sigma = 1, n){
  power.vector <- NULL
  mu <- seq(0, 2, .2)
  for(i in 1:length(mu)){

```

```

    power.sum <- 0
    for(j in 1:n){
      x <- cbind(rep(1:3, each = 16), c(rnorm(32), rnorm(16, mu[i],
sigma)))
      fit <- lm(x[, 2] ~ factor(x[, 1]))
      F.value <- anova(fit)$'F value'[1]
      power <- pf(qf(.95, 2, 45), 2, 45, ncp=F.value*2, lower.tail = F)
      power.sum <- power.sum + power
    }
    power.vector <- c(power.vector, mean(power.sum))
  }
  return(cbind(mu, power.vector))
}

```

```

constant.var <- power.test(n = 1000)
nonconstant.var <- power.test(sigma = sqrt(2), n = 1000)
plot(constant.var, type = 'l',
     main = "Compare Constant and Non-Constant Variance with Power in
simu 1000 times")
lines(nonconstant.var, lty = 2)
legend('topleft', lty = 1:2, legend = c("Constant variance", "Non-
Constant variance"))

```

```

#3#####

```

```

func.a <- function(x){
  x^3 + 2*(x^2) + 3*x - 1
}
func.b <- function(x){
  exp(1) - 1/(3.5 + x)
}
func.c <- function(x){
  exp(-x)/sqrt(1 + x^2) - 0.5
}

```

```

bisection <- function(f, a, b){
  h <- abs(b - a)/10
  i <- 0
  j <- 0
  a1 = b1 = 0
  while(i < 10){
    a1 = a + i * h
    b1 = a1 + h
    if(f(a1) == 0){
      cat('Root:', a1, 'n:', j+1)
    }else if(f(b1) == 0){
      cat('Root:', b1, 'n:', j+1)
    }else if(f(a1) * f(b1) < 0){
      repeat{
        if(abs(b1 - a1) < 1e-6){break}
        c <- (a1 + b1)/2
        if(f(a1) * f(c) < 0){
          b1 <- c

```



```

        j <- j+1
      }else{
        a1 <- c
        j <- j+1
      }
    }
    c <- (a1 + b1)/2
    cat('Root:', c, 'n:', j+1, 'f:', f(c), '\n')
  }
  i <- i+1
}
if(j == 0){
  cat("there is no root between", a, "and", b)
}
}

false.posi <- function(f, init1, init2, maxiter=1000, tol=1e-06) {
  init1[2] <- f(init1)
  init2[2] <- f(init2)
  if (init1[2] == 0.0) {return(init1)}
  if (init2[2] == 0.0) {return(init2)}
  for (i in 1:maxiter) {
    dummy <- init2[1]
    init2[1] <- init1[1]-init1[2]*(init1[1]-init2[1])/(init1[2]-
init2[2])
    init2[2] <- f(init2[1])
    if (abs(init2[1]-dummy) < tol) break
  }
  list(root = init2[1], f.root = init2[2], iter = i)
}

curve(func.a, xlim = c(-3,3), lwd = 2)
abline(h=0, lty = 2)
points(0.275, 0, cex = 2, pch = 21, bg = 1)
bisection(func.a, -4, 3)
false.posi(func.a, -4, 3)
uniroot(func.a, c(-4, 3))

curve(func.b, xlim = c(-5,-2), lwd = 2)
abline(h=0, lty = 2)
points(-3.132, 0, cex = 2, pch = 21, bg = 1)
bisection(func.b, -3.4, -2)
false.posi(func.b, -3.4, -2)
uniroot(func.b, c(-3.4, -2))

curve(func.c, xlim = c(-2,2), lwd = 2)
abline(h=0, lty = 2)
points(0.558, 0, cex = 2, pch = 21, bg = 1)
bisection(func.c, -1, 2)
false.posi(func.c, -1, 2)
uniroot(func.c, c(-1, 2))

```

```

#4#####
mitinom.loglike <- function(theta){
  1997/(2+theta) - 906/(1-theta) - 904/(1-theta) + 32/(theta)
}
curve(mitinom.loglike, from=-0.3, to=0.3)
abline(h=0)

secant <- function(f, init1, init2, maxiter=1000, tol=1e-06){
  init1[2] <- f(init1)
  init2[2] <- f(init2)
  if (init1[2] == 0.0) {return(init1)}
  if (init2[2] == 0.0) {return(init2)}
  for (i in 1:maxiter) {
    dummy <- init2
    init2[1] <- init1[1]-init1[2]*(init1[1]-init2[1])/(init1[2]-
init2[2])
    init2[2] <- f(init2[1])
    init1 <- dummy
    if (abs(init2[1]-init1[1]) < tol) break
  }
  list(root = init2[1], f.root = init2[2], iter = i)
}

ridder <- function(f, init1, init2, maxiter=1000, tol=1e-06){
  init1[2] <- f(init1)
  init2[2] <- f(init2)
  if (init1[2] == 0.0) {return(init1)}
  if (init2[2] == 0.0) {return(init2)}
  if (sign(init1[2]*init2[2]) != -1)
    stop("f() values at end points not of opposite sign")
  for (i in 1:maxiter) {
    init3 <- (init1[1] + init2[1])/2
    init3[2] <- f(init3)
    init4 <- init3[1] + (init3[1]-init1[1])*sign(init1[2])*init3[2]/
sqrt(init3[2]^2-init1[2]*init2[2])
    init4[2] <- f(init4)
    if (sign(init3[2]*init4[2]) == -1){init1 <- init3}
    else if (sign(init1[2]*init4[2]) == -1){init1 <- init1}
    else if (sign(init2[2]*init4[2]) == -1){init1 <- init2}
    else {stop("f() values at iteration points not of opposite sign")}
    init2 <- init4
    if (abs(init2[1]-init1[1]) < tol) break
  }
  list(root = init2[1], f.root = init2[2], iter = i)
}

newton Raphson <- function(f, init, maxiter=1000, tol=1e-06){
  require(numDeriv)
  if (f(init) == 0.0) {return(init)}
  for (i in 1:maxiter) {
    dx <- genD(func = f, x = init)$D[1]
    init.next <- init - f(init)/dx
    if (abs(init.next - init) < tol) break
  }
}

```

```

    init <- init.next
  }
  list(root = init.next, f.root = f(init.next), iter = i)
}

uniroot(mitinom.loglike, c(0.01, 0.99), tol = 1e-06)
secant(mitinom.loglike, .01, .99)
ridder(mitinom.loglike, .01, .99)
newton.raphson(mitinom.loglike, .01)

#5#####

wide <- read.csv("StatisticalSimulation/maleddeathrates.csv", header =
T)
long_p <- reshape(wide, direction = "long",
                  varying = list(names(wide)[2:4]),
                  v.names = "p", idvar = "year",
                  timevar = "t", times = 1:3)
long_n <- reshape(wide, direction = "long",
                  varying = list(names(wide)[5:7]),
                  v.names = "n", idvar = "year",
                  timevar = "t", times = 1:3)
long_d <- reshape(wide, direction = "long",
                  varying = list(names(wide)[8:10]),
                  v.names = "d", idvar = "year",
                  timevar = "t", times = 1:3)
long <- cbind(long_p, long_n, long_d)[, c(1, 8, 9, 18, 27)]
long <- long[order(long$year),]
write.table(long, file = "StatisticalSimulation/maleddeathrates2.CSV",
            sep="," , row.names = F, na = "NA")

long <- read.csv("StatisticalSimulation/maleddeathrates2.csv", header =
T)
p <- 1-long$p/1000
llp <- log(-log(p))
d <- long$d
x <- long$year
n <- as.numeric(long$n)

wls <- function(par) {
  x1 <- par[1]
  x2 <- par[2]
  sum(n*(llp-x1-x2*x)^2) }
root <- nlminb(start=c(0, 0), obj = wls)
(c <- exp(root[[1]][2]))
(b <- exp(root[[1]][1] + log(log(c)) - log(c - 1)))
15078.4/301

nlm <- function(par) {
  x1 <- par[1]
  x2 <- par[2]
  sum(n*(p-exp(-x1*(x2^x)*(x2-1)/log(x2)))^2) }
nlminb(start=c(0.01, 1.04), obj = nlm)

```

```
mle <- function(par) {  
  x1 <- par[1]  
  x2 <- par[2]  
  sum(((n-d)*x1*(x2^x)*(x2-1)/log(x2)-d*log(1-exp(-x1*(x2^x)*(x2-1)/log(x2))))^2}  
  nlminb(start=c(0.01, 1.04), obj = mle)
```