统计软件 HW11

邵智轩 1400012141 物理学院

贮存可靠性评估问题

```r
reliab <- read.csv("store-reliab-data.csv")
Z1 <- reliab[reliab['testid'] == 1, c('times', 'delta')]
theta <- reliab[reliab['testid'] == 1, 'true_theta'][1] # 数据中的 true theta 都是 10

P_theta_Z <- function(P, delta)
  prod(ifelse(delta, P, 1 - P))
```

直接求值法

```r
G.direct <- function(theta, Z0) {
  N <- nrow(Z0)
  P <- exp(-Z0[, 1] / theta)
  n_0 <- sum(Z0[, 2])
  G.sum <- 0
  # 满足条件 1 的 Zn
  if (n_0 < N) {
    for (n in (n_0 + 1):N) {
      delta.combn <- combn(N, n)
      G.temp <- vapply(1:ncol(delta.combn), function(i) {
        delta1 <- rep(0, N)
        delta1[delta.combn[, i]] <- 1
        return (P_theta_Z(P, delta1))
      }, FUN.VALUE = 1)
      G.sum <- G.sum + sum(G.temp)
    }
  }
  # 满足条件 2 的 Zn （n0==N）
  delta.combn <- combn(N, n_0)
  G.temp <- vapply(1:ncol(delta.combn), function(i) {
    delta1 <- rep(0, N)
    delta1[delta.combn[, i]] <- 1
    if (sum(Z0[, 1] * delta1) >= sum(Z0[, 1] * Z0[, 2]))
      return (P_theta_Z(P, delta1))
    else
      return (0)
  }, FUN.VALUE = 1)
  G.sum <- G.sum + sum(G.temp)
  G.sum
}
G.direct <- Vectorize(G.direct, vectorize.args = "theta") # 向量化
```

随机模拟法

```
G.MC <- function(theta, Z0, N = 1e4) {
  n0 <- sum(Z0[, 2])
  G.MC.single <- function(theta, Z0) {
    # 返回 I(Zn(i)>Zn)
    X <- rexp(nrow(Z0), 1 / theta)
    delta <- X > Z0[, 1]
    # 满足条件 1 的 Zn
    if (sum(delta) > n0)
      return (TRUE)
    if (sum(delta) == n0) {
      if (sum(delta * Z0[, 1]) >= sum(Z0[, 2] * Z0[, 1]))
        return (TRUE)
    }
    return (FALSE)
  }
  x <- replicate(N,G.MC.single(theta, Z0))
  c(mean = mean(x), sd = sd(x))
}
G.MC <- Vectorize(G.MC, vectorize.args = "theta")
```

试验两方法求解出的 $\theta$ 是否相同

```
# 对 testid 为 1 的数据进行试验，选取不同的 $\theta$
ptm <- proc.time()
G.direct(10, Z1)
```

```
## [1] 0.8880072
```

```
proc.time() - ptm
```

```
##    user  system elapsed
##    0.01    0.00    0.02
```

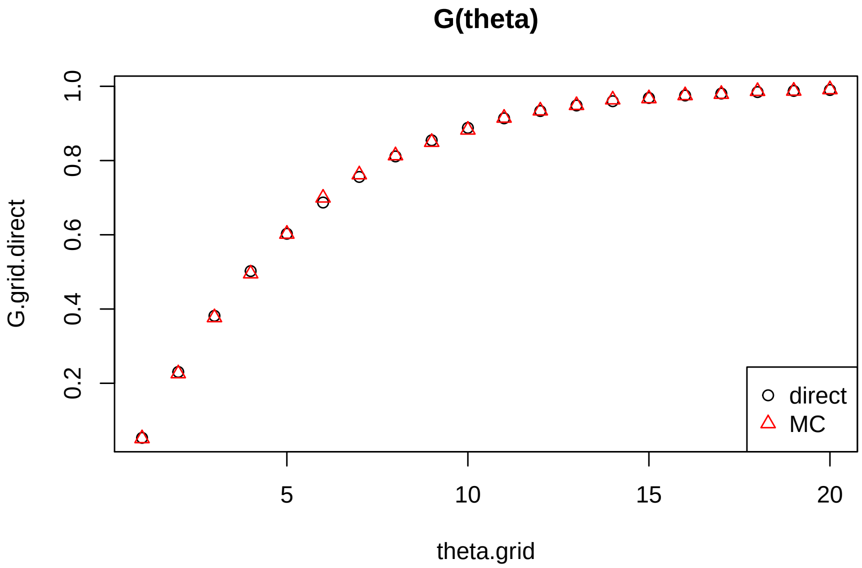
```
ptm <- proc.time()
G.MC(10, Z1, 1e4)
```

```
##            [,1]
## mean 0.8952000
## sd   0.3063109
```

```
proc.time() - ptm
```

```
##    user  system elapsed
##    0.35    0.00    0.34
```

```
# 两种方法作 G(theta) 图, 并对比
theta.grid <- 1:20
G.grid.direct <- G.direct(theta.grid, Z0 = Z1)
plot(theta.grid, G.grid.direct,main = "G(theta)")
G.grid.MC <- G.MC(theta.grid, Z0 = Z1)
points(theta.grid, G.grid.MC[1,],col="red",pch=2)
legend("bottomright",c("direct","MC"),
       pch=c(1,2),col=c("black","red"))
```



随机模拟不同于随机误差。可以看到在 $n = 8$ 使，如果随机模拟的次数设得较高，反而比直接求解慢。但当 n 变大时，直接求解得时间复杂度指数上升，随机模拟法只是线性上升，优势就体现出来了，如下面的试验：

```
Z2 <- reliab[reliab['testid'] <= 2, c('times', 'delta')]
ptm <- proc.time()
G.direct(10, Z2)
```

```
## [1] 0.640969
```

```
proc.time() - ptm
```

```
##    user  system elapsed
##    0.98    0.00    0.99
```

```
ptm <- proc.time()
G.MC(10, Z2, 1e4)
```

```
##            [,1]
## mean 0.6395000
## sd   0.4801696
```

```
proc.time() - ptm
```

```
##    user  system elapsed
##    0.27    0.00    0.26
```

可以看到，$n = 16$ 时，MC 方法得效率优势已经体现。

求解方程 $G(\theta) - \alpha = 0$（$\alpha = 0.05$）

直接法求 G，二分法求根

```
solve.theta <- function(alpha,
                        Z0,
                        G = function(theta)
                          G.direct(theta, Z0),
                        theta.min = 0,
                        theta.max = 50,
                        eps = 1e-5) {
  if (sum(Z0==1)==0) return(0) # 所有 n 次试验都失效的情形
  # 二分法求根
  if (alpha >= 1) {
    print ("Illegal value of alpha!")
  }
  if (G(theta.max) < alpha) {
    print("theta.max is not large enough!")
    return (FALSE)
  }
  while (abs(theta.max - theta.min) > eps) {
    theta.new <- (theta.min + theta.max) / 2
    if (G(theta.new) > alpha)
      theta.max <- theta.new
    else
      theta.min <- theta.new
  }
  (theta.min + theta.max) / 2
}

# 直接法求 G(theta)
ptm=proc.time()
solve.theta(0.05, Z1)
```

```
## [1] 0.9813935
```

```
proc.time()-ptm
```

```
##    user  system elapsed
##    0.12    0.00    0.13
```

```
# 随机模拟法求 G(theta)
ptm=proc.time()
solve.theta(
  0.05,
  Z1,
  G = function(theta)
    G.MC(theta, Z1)[1]
)
```

```
## [1] 1.001075
```

```
proc.time()-ptm
```

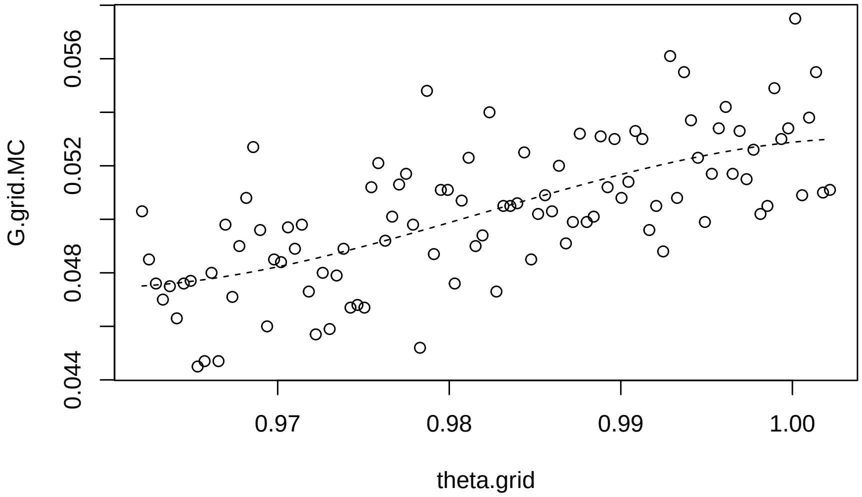
```
##    user  system elapsed
##    5.13    0.00    5.14
```

可以看到，随机模拟法由于每次计算有随机误差，用 naive 得二分法求根结果不够准确，需要对求根算法做一些改进。下面是利用样条回归改进后的算法：

```
solve.theta <- function(alpha,
                        Z0,
                        G.method = "direct",
                        N = NULL, spline.plot =TRUE,
                        theta.min = 0,
                        theta.max = 50,
                        eps = 1e-4) {
  if (sum(Z0==1)==0) return(0) # 所有 n 次试验都失效的情形
  # 二分法求根
  if (alpha >= 1) {
    print ("Illegal value of alpha!")
    return(FALSE)
  }
  if (G.method=="direct"){
    G<-function(theta) G.direct(theta, Z0)
  }
  else if (G.method=="MC"){
    G<-function(theta) G.MC(theta,Z0,ifelse(is.null(N),1e4,N))[1]
  }
  else {
    print("Illegal G.method!")
    return(FALSE)
  }
  if (G(theta.max) < alpha) {
    print("theta.max is not large enough!")
    return (FALSE)
  }
  while (abs(theta.max - theta.min) > eps) {
    theta.new <- (theta.min + theta.max) / 2
    if (G(theta.new) > alpha)
      theta.max <- theta.new
    else
      theta.min <- theta.new
  }
  if (G.method=="direct") (theta.min + theta.max) / 2
  else if (G.method=="MC"){
    # 并不直接返回，而是在其领域内做样条回归，在线上找最接近 alpha 的点
    theta.grid<-seq(theta.min-200*eps,theta.max+200*eps,length.out = 100)
    G.grid.MC<-vapply(theta.grid,G,FUN.VALUE = 1)
    require(splines)
    reg.spline.fitted<-lm(G.grid.MC~bs(theta.grid))$fitted.values
    if (spline.plot){
      plot(theta.grid,G.grid.MC)
      lines(theta.grid,reg.spline.fitted,lty="dashed")
    }
    theta.grid[which.min(abs(reg.spline.fitted-alpha))]
  }
  #(theta.min + theta.max) / 2
}

ptm=proc.time()
solve.theta(0.05, Z1, G.method = "MC")
```

```
## Loading required package: splines
```



```
## [1] 0.980724
```

```
proc.time()-ptm
```

```
##    user  system elapsed
##   24.30    0.01   24.44
```

可以看到，用改进后的算法虽然会花费更多时间，但是很大程度上减小了随机误差造成的影响，更加准确。

求解 store-reliab-data.csv 中问题

```
ptm=proc.time()
result <- vapply(1:10, function(testid) {
  Z <- reliab[reliab['testid'] == testid, c('times', 'delta')]
  theta <- reliab[reliab['testid'] == testid, 'true_theta'][1]
  alpha <- reliab[reliab['testid'] == testid, 'alpha'][1]
  c(direct = solve.theta(alpha, Z),
    MC = solve.theta(alpha, Z, G.method = "MC", spline.plot = F))
}, FUN.VALUE = c(0.5, 0.5))
proc.time()-ptm
```

```
##    user  system elapsed
##  235.88    0.05  236.81
```

```
result
```

```
##              [,1]     [,2]     [,3]     [,4]     [,5] [,6]     [,7]
## direct 0.9813786 5.269861 4.315329 4.904222 26.02706    0 15.64975
## MC     0.9832704 5.265162 4.285385 4.891751 26.54250    0 15.63808
##            [,8]     [,9]    [,10]
## direct 2.610636 5.088663 4.026270
## MC     2.624348 5.078317 4.037358
```