品質管制 Homework 5

110024516 統研碩一邱繼賢

2021 年 10 月 30 日

4.5

設定變數 $\rho = 0.5,\ [h_l,\ h_u] = [0,\ 15],\ maximum\ number\ of\ iterations : m = 50$

然後定義出 $compute\_h\ function$ 用來計算 h 的數值，由於本題的 k 值皆 ≤ 1，故在計算每次迴圈 ARL_0 數值時可直接使用 Siegmund 的公式：

$$ARL_0 \approx \frac{exp(2k(h+1.166)) - 2k(h+1.166) - 1}{2k^2}$$

並檢查迴圈進行時，計算出的 ARL_0 是否 $\in [A_0-\rho, A_0+\rho](ps : A_0$ 即為以下三小題的 $ARL_0)$，若結果為是，則函數回傳當時的 h 值即為所求。

```
rho = 0.5
h_l = 0
h_u = 15
m = 50

compute_h = function(arl_0, k) {
    for (i in 1:m) {
        h = (h_l + h_u)/2
        arl = (exp(2*k*(h+1.166))-2*k*(h+1.166)-1)/(2*k^2)
        if (arl>(arl_0-rho) & arl<(arl_0+rho)) {
            break
        } else if (arl > arl_0) {
            h_u = h
        } else if (arl < arl_0) {
            h_l = h
        }
    }
    return(h)
}
```

(i) $ARL_0 = 350,\ k = 0.25$

```
compute_h(350, 0.25)
```

```
## [1] 6.602783
```

$h \approx 6.6028$

(ii) $ARL_0 = 350, \ k = 0.5$

```
compute_h(350, 0.5)
```

```
## [1] 4.033813
```

$h \approx 4.0338$

(iii) $ARL_0 = 550, \ k = 0.25$

```
compute_h(550, 0.5)
```

```
## [1] 4.475098
```

$h \approx 4.4751$

結論：

(1) 根據 (i) 和 (ii) 計算出的數值，可以得知，在 ARL_0 相同的情況下，k 和 h 呈現負相關。

(2) 根據 (ii) 和 (iii) 計算出的數值，可以得知，在 k 相同的情況下，ARL_0 和 h 呈現正相關。

以上兩點結論皆可以從以下此公式推得

$$ARL_0 \approx \frac{exp(2k(h+1.166)) - 2k(h+1.166) - 1}{2k^2}$$

4.9

定義兩個函數 *compute_arl* 和 *compute_arl_2*，前者為使用模擬資料和迴圈來計算出 ARL_0 的方法，適用於任意的 k 值，但因為設計成雙層迴圈，故計算速度會較慢；後者為直接使用 Siegmund 的公式計算出 ARL_0 的方法，只適用於 $k \le 1$ 時的情況，但計算速度也較快。

另定義兩函數 *compute_k_star* 和 *compute_h_star*，用來計算 $k^\star = \frac{k-\delta}{\lambda\sigma}$ 和 $h^\star = \frac{h}{\lambda\sigma}$, *where the process distribution changes from* $N(\mu_0\ ,\ \sigma^2)$ *to* $N(\mu_0+\delta\ ,\ \lambda^2\sigma^2)$

將新計算好的 $k^\star$ 和 $h^\star$ 代入 *compute_arl* 進行模擬計算出 ARL_1，或是代入 *compute_arl_2* 使用 Siegmund 公式：

$$ARL_1 \approx \frac{exp(2k^\star(h^\star+1.166)) - 2k^\star(h^\star+1.166) - 1}{2k^{\star 2}}$$

```r
k = 0.5
h = 4.095
M = 100000
rl = c()
compute_arl = function(k_star, h_star) {
    for (i in 1:M) {
        C_n = c()
        X_1 = rnorm(1)
        C_n[1] = max(0, 0-X_1-k_star)
        if (C_n[1] > h_star) {
            rl[i] = 1
            next
        }
        j = 2
        while (T){
            C_n[j] = max(0, C_n[j-1]-rnorm(1)-k_star)
            if (C_n[j] > h_star) {
                rl[i] = j
                break
            }
            j = j+1
        }
    }
    return(mean(rl))
}

compute_arl_2 = function(k_star, h_star) {
    return((exp(2*k_star*(h_star+1.166))-2*k_star*(h_star+1.166)-1)/(2*k_star^2))
}

compute_k_star = function(delta, lambda) {
    return((k-delta)/(lambda*1))
}

compute_h_star = function(lambda) {
    return(h/(lambda*1))
}
```

(i) $N(0.5,\ 1)$
$\delta\ =\ 0.5,\ \lambda\ =\ 1\ \Rightarrow\ k^{\star}\ =\ 0,\ h^{\star}\ =\ 4.095$

此題 $k^{\star}\ =\ 0$ 代入 $compute\_arl\_2$ 函數的公式會造成分母為零的情況，故只能使用 $compute\_arl$ 函數計算，ARL_1 計算結果如下：

```
k_star1 = compute_k_star(0.5, 1)
h_star1 = compute_h_star(1)
compute_arl(k_star1, h_star1)
```

```
## [1] 27.76373
```

(ii) $N(1,\ 1)$
$\delta\ =\ 1,\ \lambda\ =\ 1\ \Rightarrow\ k^{\star}\ =\ -0.5,\ h^{\star}\ =\ 4.095$

此題使用兩種函數計算 ARL_1 結果如下：

```
k_star2 = compute_k_star(1, 1)
h_star2 = compute_h_star(1)
compute_arl(k_star2, h_star2)
```

```
## [1] 8.57674
```

```
compute_arl_2(k_star2, h_star2)
```

```
## [1] 8.53238
```

(iii) $N(-1,\ 1)$
The both methods of calculating ARL_1 are designed for detecting an upward shift. In order to detect a downward shift of size $-\delta(\delta\ >\ 0)$ in quality characteristic X_i is equivalent to detect an upward shift of size δ in $-X_i$ whose process distribution is $N(1,\ 1)$

$\Rightarrow\ \delta\ =\ 1,\ \lambda\ =\ 1\ \Rightarrow\ k^{\star}\ =\ -0.5,\ h^{\star}\ =\ 4.095$

In this problem, we have calculated ARL_1 with the both methods as below:

```
k_star3 = compute_k_star(1, 1)
h_star3 = compute_h_star(1)
compute_arl(k_star3, h_star3)
```

```
## [1] 8.55865
```

```
compute_arl_2(k_star3, h_star3)
```

```
## [1] 8.53238
```

(iv) $N(0,\ 2^2)$
$\delta\ =\ 0,\ \lambda\ =\ 2\ \Rightarrow\ k^{\star}\ =\ 0.25,\ h^{\star}\ =\ 2.0475$

此題使用兩種函數計算 ARL_1 結果如下：

```
k_star4 = compute_k_star(0, 2)
h_star4 = compute_h_star(2)
compute_arl(k_star4, h_star4)
```

```
## [1] 18.98754
```

```
compute_arl_2(k_star4, h_star4)
```

```
## [1] 19.03863
```

(v) $N(0,\ 0.5^2)$
$\delta\ =\ 0,\ \lambda\ =\ 0.5\ \Rightarrow\ k^{\star}\ =\ 1,\ h^{\star}\ =\ 8.19$

此題因為 *variance* 變小，而且 *mean* 沒有改變，故計算出的 ARL_1 數值會非常大，使用 *compute_arl* 函數計算時間會太久，所以只呈現 *computa_arl_2* 函數計算出的 ARL_1 數值如下：

```
k_star5 = compute_k_star(0, 0.5)
h_star5 = compute_h_star(0.5)
compute_arl_2(k_star5, h_star5)
```

```
## [1] 66909577
```

(vi) $N(1,\ 2^2)$
$\delta\ =\ 1,\ \lambda\ =\ 2\ \Rightarrow\ k^{\star}\ =\ -0.25,\ h^{\star}\ =\ 2.0475$

此題使用兩種函數計算 ARL_1 結果如下：

```
k_star6 = compute_k_star(1, 2)
h_star6 = compute_h_star(2)
compute_arl(k_star6, h_star6)
```

```
## [1] 6.48828
```

```
compute_arl_2(k_star6, h_star6)
```

```
## [1] 6.458306
```

結論：

(1) 根據 (i) 和 (ii) 以及 (iv) 和 (vi) 的兩兩比較，可以得知，在 *variance shift* 程度相同時，*positive mean shift* 程度越大者，ARL_1 則越小。

(2) 根據 (ii) 和 (vi) 以及 (iv) 和 (v) 的兩兩比較，可以得知，在 *positive mean shift* 程度相同時，*variance shift* 程度越大者，ARL_1 則越小。

(3) 根據 (ii) 和 (iii) 兩組比較，可以得知，兩者的 *mean shift* 的程度大小相同，只不過偏移方向相反（前者為正、後者為負）情況下，兩者所計算出的 ARL_1 數值差異不大。