# Statistical Computing and Simulation: Assignment 4

Deparment of Statistics, NCCU
葉佐晨　高崇哲
{112354016,112354020}@nccu.edu.tw

2024-05-16

## Statistical Computing and Simulation

### Assignment 4, Due May 17/2024

### Question 1

Suppose the central death rate $m_x = 0.04$. Compute the mortality rate $q_x$ under the uniform death distribution, constant force, and hyperbolic assumption.

從資料來源中，可以得到各種假設下$m_x$和$q_x$的關係式，接著各定義一組函數，用來計算$m_x$和$q_x$之間的平方差，然後使用nlminb函數找固定 $m_x$下，最小化平方差的$q_x$。

資料來源: https://users.stat.ufl.edu/~rrandles/sta4930/4930lectures/chapter3/chapter3R.pdf

#### Hyperbolic

```
## $par
## [1] 0.03920544
##
## $objective
## [1] 5.071929e-24
##
## $convergence
## [1] 0
##
## $iterations
## [1] 3
##
```

```
## $evaluations
## function gradient
##        5        4
##
## $message
## [1] "X-convergence (3)"
```

**UDD**

```
## $par
## [1] 0.03921569
##
## $objective
## [1] 4.756966e-24
##
## $convergence
## [1] 0
##
## $iterations
## [1] 3
##
## $evaluations
## function gradient
##        5        4
##
## $message
## [1] "X-convergence (3)"
```

**Constant Force**

```
## $par
## [1] 0.03921056
##
## $objective
## [1] 4.889144e-24
##
## $convergence
## [1] 0
##
## $iterations
```

```
## [1] 3
##
## $evaluations
## function gradient
##        5        4
##
## $message
## [1] "X-convergence (3)"
```

從估計結果來看，表現最好的是Hyperbolic，再來是Constant Force，最後是 UDD，迭代次數在每種假設下均為三次。

## Question 2

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 2019-2021. 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.

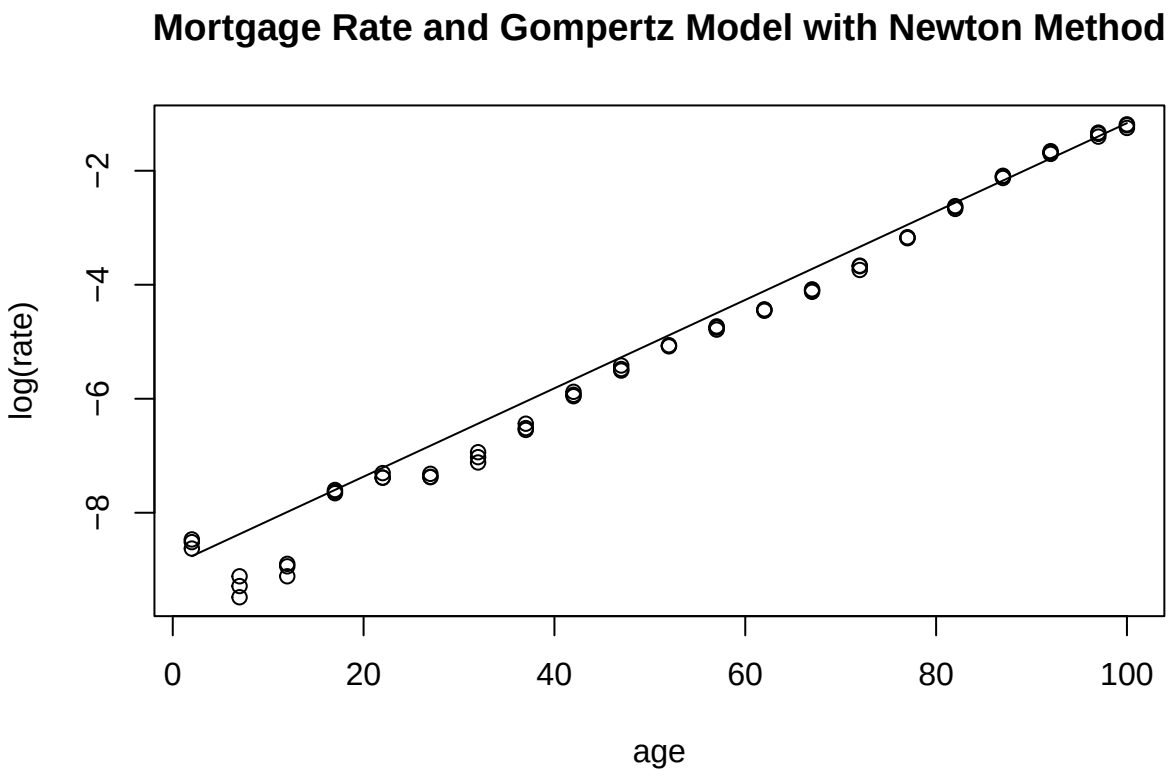
**Solution**

表 1: Age Mortgage Rate in 2019-2021

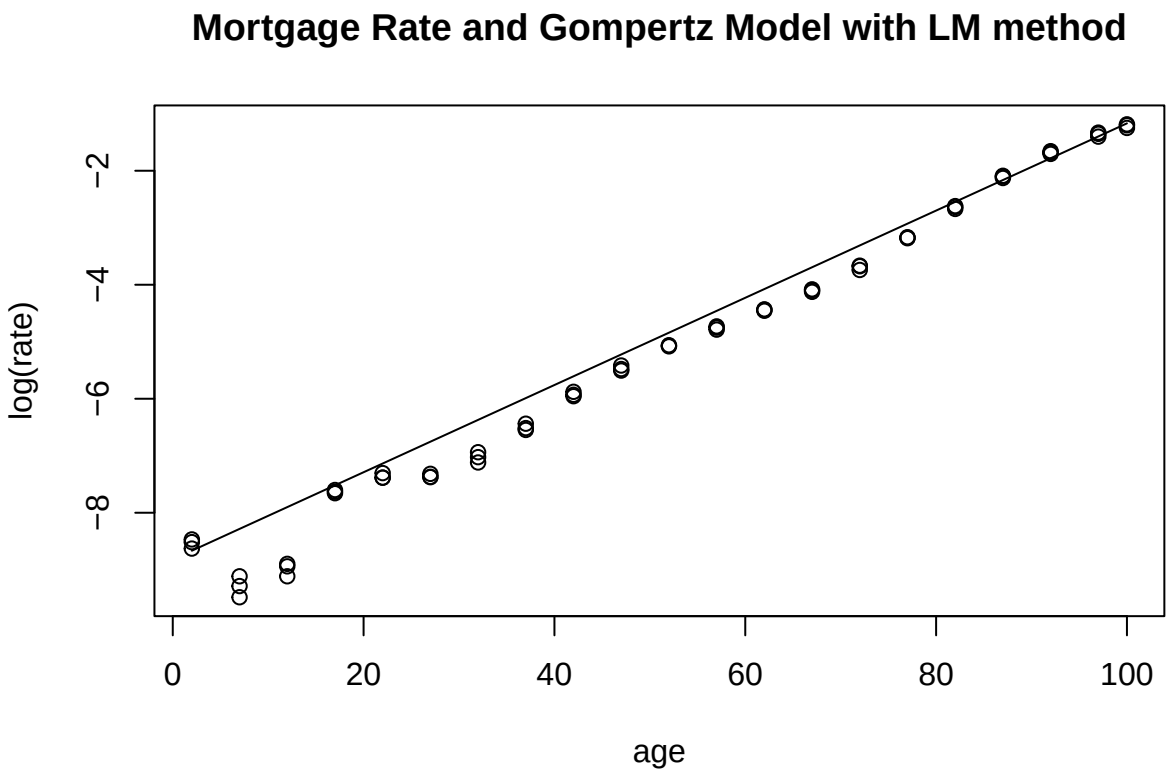
| year | X0 | X2 | X7 | X12 | X17 | X22 | X27 | X32 | X37 | X42 | X47 | X52 | X57 | X62 | X67 | X72 | X77 | X82 | X87 | X92 | X97 | X100 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2019 | 4.3 | 0.2 | 0.1 | 0.1 | 0.5 | 0.6 | 0.6 | 1.0 | 1.6 | 2.8 | 4.4 | 6.3 | 8.8 | 11.8 | 16.7 | 25.6 | 41.8 | 72.8 | 122.2 | 186.5 | 258.6 | 305.5 |
| 2020 | 4.0 | 0.2 | 0.1 | 0.1 | 0.5 | 0.6 | 0.6 | 0.8 | 1.5 | 2.6 | 4.1 | 6.2 | 8.3 | 11.6 | 16.2 | 23.7 | 41.5 | 69.3 | 119.2 | 181.8 | 246.3 | 286.8 |
| 2021 | 4.6 | 0.2 | 0.1 | 0.1 | 0.5 | 0.7 | 0.7 | 0.9 | 1.4 | 2.7 | 4.2 | 6.3 | 8.6 | 11.9 | 16.9 | 25.3 | 42.2 | 71.6 | 123.9 | 190.9 | 264.3 | 300.3 |

上表是各年齡組的死亡率（千分比），使用組中點做為年齡$x$，使用Gompertz model估計B和C。分別使用`nls`的Newton method、Levenberg-Marquardt Method、Steepest Descent進行估計，再來比較B和C的估計值、誤差平方和、以及迭代次數。由於可以得到B和C的估計值，因此可以比較樣本以及使用模型預估的死亡率，並用散布圖加上迴歸線呈現。

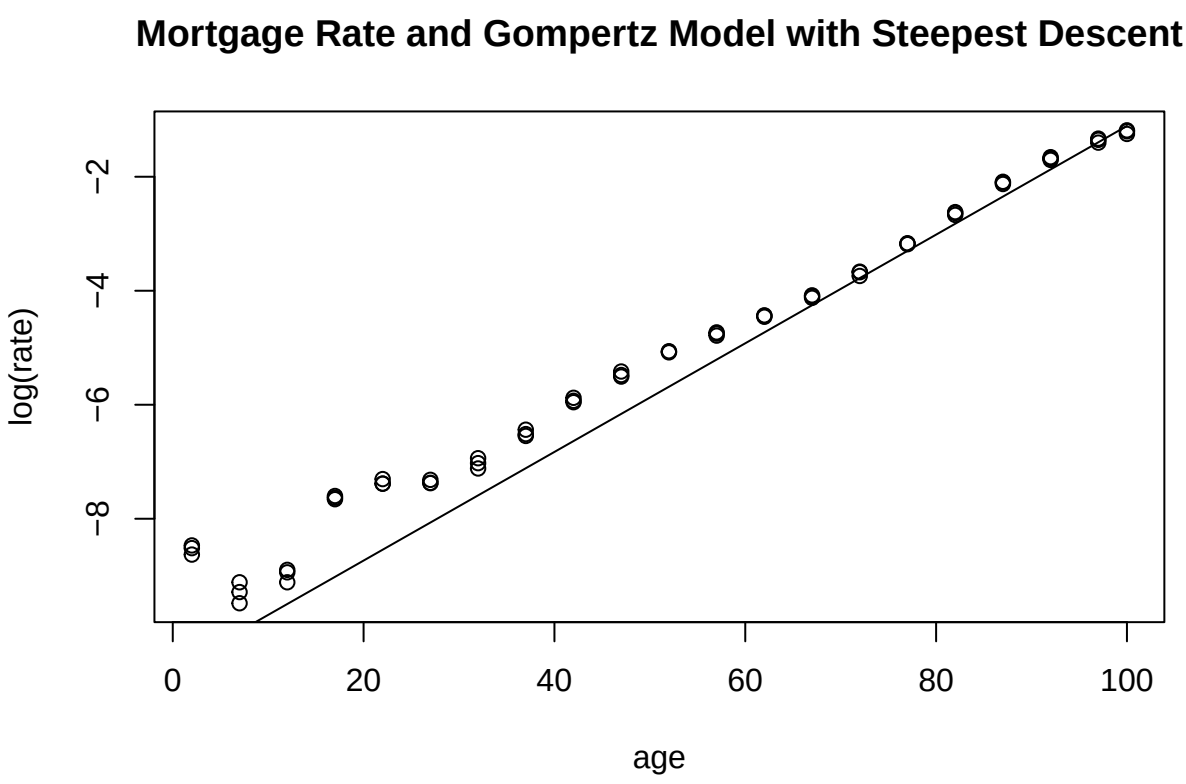
**Newton Method with nls** 在Newton Method中，使用`nls`估計Gompertz model $\mu_x = BC^x$中的係數，並使用圖表呈現。其中B的估計值是0.00013，C的估計值是1.08061。




**Levenberg-Marquardt Method** 使用Levenberg-Marquardt Method估計的B為0.00015，而C為1.07951，可以看到估計值和Newton method的結果相差不大，不過迭代次數是51次，是三者方法中次數最多的。




**Steepest Descent** Steepest Descent估計的B是0.00002，C是1.1，C的估計值和一開始提供的起始點相同，表示在C方向上並未進行更新。Steepest Descent的估計誤差比前兩者稍大，不過迭代次數只有2次，估計速度相當快。




**Summary** 下表是三個估計方法的估計結果，綜合來說LM method的估計效果最好，不過迭代次數相當高，總共迭代了51次，比其他兩者的高很多。而Steepest Descent的迭代次數最少，可是估計值和其他兩者相當不同，更有可能是收斂在局部最小的部位。

以起始點討論，Newton method和LM method對起始點較不敏感，在合理的參數範圍內都可以收斂到相同的位置。而Steepest Descent對起始值相當敏感，只有一開始就在local minimum附近的位置才會收斂。

表 2: Summary of three Methods of Optimization

| | B.est | C.est | SSE | iter |
|---|---|---|---|---|
| Newton | 0.00013 | 1.08061 | 0.00767 | 16 |
| LM | 0.00015 | 1.07951 | 0.00361 | 51 |
| SteepDesc | 0.00002 | 1.10000 | 0.00964 | 2 |

## Question 3

Consider a multinomial observation $X = (x_1, x_2, x_3, x_4)$ with class probabilities given by $(p_1, p_2, p_3, p_4) = (\frac{2+\theta}{4}, \frac{1-\theta}{4}, \frac{1-\theta}{4}, \theta)$, 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-Raphosn methods to find the MLE (via $l'(\theta)$). You may choose your own starting points and convergence criterion (preferred $10^{-6}$ or smaller).

先定義概似函數及其導數，再使用三種方法分別進行估計。

1. Newton method:透過概似函數值及導數值迭代，收斂速度快，對初始值選擇敏感。
2. Secant method:透過概似函數值及兩個初始值迭代，計算方便，收斂速度慢。
3. Ridder method:類似於割線法，但會透過概似函數的符號變化來修正逼近值。

每個方法可以制定初始值和誤差門檻，最後結果為估計值、概似函數值以及迭代次數。

**Newton method**

```
##          ans   Likelihood iteration
## 1 0.03571241 -0.002994194         7
```

**Secant method**

```
##         ans   Likelihood iteration
## 1 0.0357123 2.470836e-08        14
```

**Ridder method**

```
##         ans    Likelihood iteration
## 1 0.0357123 -3.410605e-13         5
```

三種方法的估計值差異不大，只有Newton method略為較大，從迭代次數來比較，使用最少次迭代的是Ridder method，再來是Newton method，最後是Secant method。

## Question 4

Evaluate the CDF of standard normal distribution $\Phi(x)$ using the method of Important Sampling and other Variance Reduction Methods (at least two different methods). Consider $x = `6, `5, `4, `3.5, `3, `2.5, `2.

**Solution**

我們在這裡使用Important Sampling和Monte-Carlo Integration估計$\Phi(x)$。每個估計式都分別使用了100、1000、5000個樣本計算估計值，在不同的樣本數下各計算1000個估計值，再來比較1000個估計值的平均數，以此來比較不同方法中，不同的$x$ quantile以及不同樣本數的的情況下的估計表現，並且和理論值比較。

**Important Sampling**　在Important Sampling中，在各種樣本數以及$x$的情況下，估計值都相當接近理論值。

表 3: Mean of Estimates with Important Sampling

| | n=100 | n=1000 | n=5000 | pnorm |
|---|---|---|---|---|
| x=-6 | 0.0000000010 | 0.0000000010 | 0.0000000010 | 0.0000000010 |
| x=-5 | 0.0000002874 | 0.0000002870 | 0.0000002866 | 0.0000002867 |
| x=-4 | 0.0000316922 | 0.0000316441 | 0.0000316724 | 0.0000316712 |
| x=-3.5 | 0.0002325811 | 0.0002327087 | 0.0002325357 | 0.0002326291 |
| x=-3 | 0.0013539367 | 0.0013507110 | 0.0013498715 | 0.0013498980 |
| x=-2.5 | 0.0062149705 | 0.0062111038 | 0.0062072256 | 0.0062096653 |
| x=-2 | 0.0227269231 | 0.0227383803 | 0.0227486829 | 0.0227501319 |

**Monte Carlo Integration**　在Monte Carlo Integration中，隨著樣本數越大，估計值的平均數有更接近理論值的傾向。

表 4: Mean of Estimates with Monte-Carlo Integration

| | n=100 | n=1000 | n=5000 | pnorm |
|---|---|---|---|---|
| x=-6 | 0.0000000010 | 0.0000000010 | 0.0000000010 | 0.0000000010 |
| x=-5 | 0.0000002913 | 0.0000002872 | 0.0000002867 | 0.0000002867 |
| x=-4 | 0.0000317514 | 0.0000317662 | 0.0000316399 | 0.0000316712 |
| x=-3.5 | 0.0002309852 | 0.0002329059 | 0.0002326764 | 0.0002326291 |
| x=-3 | 0.0013543858 | 0.0013530614 | 0.0013478947 | 0.0013498980 |
| x=-2.5 | 0.0062456606 | 0.0062342191 | 0.0062123989 | 0.0062096653 |
| x=-2 | 0.0227542353 | 0.0227884418 | 0.0227183676 | 0.0227501319 |

**Antithetic Monte Carlo Integration**　在Antithetic Monte Carlo Integration中，隨著樣本數越大，估計值的平均數有更接近理論值的傾向，整體表現相當接近一般的Monte Carlo Integration。

表 5: Mean of Estimates with Antithetic Monte-Carlo Integration

| | n=100 | n=1000 | n=5000 | pnorm |
|---|---|---|---|---|
| x=-6 | 0.0000000010 | 0.0000000010 | 0.0000000010 | 0.0000000010 |
| x=-5 | 0.0000002856 | 0.0000002870 | 0.0000002867 | 0.0000002867 |
| x=-4 | 0.0000317505 | 0.0000316679 | 0.0000317095 | 0.0000316712 |
| x=-3.5 | 0.0002338464 | 0.0002329052 | 0.0002326446 | 0.0002326291 |

| | n=100 | n=1000 | n=5000 | pnorm |
|---|---|---|---|---|
| x=-3 | 0.0013492780 | 0.0013486724 | 0.0013494990 | 0.0013498980 |
| x=-2.5 | 0.0062077018 | 0.0062143762 | 0.0062115504 | 0.0062096653 |
| x=-2 | 0.0228486079 | 0.0227241670 | 0.0227387238 | 0.0227501319 |

**Summary of Standard Deviation** 在樣本數為100和5000的情況下，Important Sampling的標準差都比兩種Monte-Carlo Integration還要低。在兩個Monte-Carlo Integration方法中，使用Antithetic method的標準差比未使用還要低，因為Antithetic的效果相當於增加一倍樣本，因此變異數會大約呈現2倍的關係。

表 6: Standard Deviation of Estimates with n=100

| | ImportantSample | MCInteg | AntMCInteg |
|---|---|---|---|
| x=-6 | 0.0000000001 | 0.0000000004 | 0.0000000003 |
| x=-5 | 0.0000000241 | 0.0000001076 | 0.0000000711 |
| x=-4 | 0.0000022181 | 0.0000091462 | 0.0000061577 |
| x=-3.5 | 0.0000149761 | 0.0000575394 | 0.0000377320 |
| x=-3 | 0.0000733719 | 0.0002816653 | 0.0001858241 |
| x=-2.5 | 0.0002787137 | 0.0011251975 | 0.0006739954 |
| x=-2 | 0.0007848949 | 0.0033205807 | 0.0017319433 |

表 7: Standard Deviation of Estimates with n=5000

| | ImportantSample | MCInteg | AntMCInteg |
|---|---|---|---|
| x=-6 | 0.0000000000 | 0.0000000001 | 0.0000000000 |
| x=-5 | 0.0000000036 | 0.0000000149 | 0.0000000095 |
| x=-4 | 0.0000003296 | 0.0000013068 | 0.0000008626 |
| x=-3.5 | 0.0000021196 | 0.0000082381 | 0.0000052575 |
| x=-3 | 0.0000103765 | 0.0000425596 | 0.0000251953 |
| x=-2.5 | 0.0000390616 | 0.0001575019 | 0.0000939688 |
| x=-2 | 0.0001095713 | 0.0004539138 | 0.0002477492 |

## Question 5

Evaluate the following quantity by both numerical and Monte Carlo integration, and compare their errors with respect to the numbers of observations used. Also, propose at least two simulation methods

to reduce the variance of Monte Carlo integration and compare their variances.

$$\theta = \int_0^1 e^{x^2}\, dx$$

- Real Value: 使用integrate函數計算在範圍[0, 1]內的積分。
- Numerical Integration: 使用黎曼積分，將積分區域切割成多個矩形進行計算。
- Monte Carlo Integration: 通過服從0到1均勻分配的隨機樣本，帶入函數計算平均值。
- Antithetic Method: 透過補數，來減少蒙特卡羅積分的估計誤差與標準誤差。
- Stratified Sampling Method: 將積分區間分成多個小區間，並在每個子區間內均勻抽樣。

```
## $`n = 50`
##          Real Value Numerical              Monte Carlo  Antithetic
## Estimate "1.462652" "1.46256114708226"     "1.50259"    "1.447306"
## SE       ""         ""                     "0.072734"   "0.043464"
## Error    ""         "9.08529177430228e-05" "0.039938"   "0.015346"
##          Stratified Sampling
## Estimate "1.462323"
## SE       "0.02217"
## Error    "0.000329"
##
## $`n = 100`
##          Real Value Numerical              Monte Carlo  Antithetic
## Estimate "1.462652" "1.46262909421928"     "1.431144"   "1.474104"
## SE       ""         ""                     "0.043202"   "0.031683"
## Error    ""         "2.29057807223931e-05" "0.031508"   "0.011452"
##          Stratified Sampling
## Estimate "1.463873"
## SE       "0.016144"
## Error    "0.001221"
##
## $`n = 500`
##          Real Value Numerical              Monte Carlo  Antithetic
## Estimate "1.462652" "1.4626508398143"      "1.485578"   "1.471337"
## SE       ""         ""                     "0.022791"   "0.015939"
## Error    ""         "1.16018570417431e-06" "0.022926"   "0.008685"
##          Stratified Sampling
## Estimate "1.462636"
## SE       "0.007024"
```

```
## Error    "1.6e-05"
##
## $`n = 1000`
##          Real Value Numerical              Monte Carlo  Antithetic
## Estimate "1.462652" "1.46265151938376"     "1.484264"   "1.466427"
## SE       ""         ""                     "0.015304"   "0.010851"
## Error    ""         "4.80616238140996e-07" "0.021612"   "0.003775"
##          Stratified Sampling
## Estimate "1.462528"
## SE       "0.005058"
## Error    "0.000124"
```

隨著觀察值的增加，四種方法的誤差都有逐漸縮小的趨勢，除了Stratified Sampling有時會些微增加，估計表現最好到最差的依序是Numerical Integration、Stratified Sampling、Antithetic、Monte Carlo Integration，在變異數方面，Antithetic和Stratified Sampling是為了有效縮減Monte Carlo Integration的變異數，從結果可以看到在精確度跟準確度都獲得不錯的提升。

## Question 6

Let $X_i, i = 1, \cdots, 5$ be random variables, following the exponential distribution with mean 1. Consider the quantity $\theta$ defined by

$$\theta = P\left(\sum_{i=1}^{5} iX_i \geq 21.6\right)$$

. Propose at least three simulation methods to estimate $\theta$ and compare their variances.

### Solution

Because $X_i$'s are from exponential distribution with $\lambda = 1$, the statistic $\sum_{i=1}^{5} iX_i$'s expectation is

$$E\left(\sum_{i=1}^{5} iX_i\right) = \sum_{i=1}^{5} iE(X_i) = \sum_{i=1}^{5} i = 15$$

and variance is

$$Var\left(\sum_{i=1}^{5} iX_i\right) = \sum_{i=1}^{5} i^2 Var(X_i) = \sum_{i=1}^{5} i^2 = 55$$

So considering Central Limit Theorem,

$$\frac{\sum_{i=1}^{5} iX_i - E\left(\sum_{i=1}^{5} iX_i\right)}{\sqrt{Var\left(\sum_{i=1}^{5} iX_i\right)}} \longrightarrow Z \sim N(0,1), \text{ as } n \to \infty$$

Therefore,

$$\theta = P\left(\sum_{i=1}^{5} iX_i \geq 21.6\right) \longrightarrow P\left(Z \geq \frac{21.6 - 15}{55}\right) = P(Z \geq 0.89)$$

So we transform the problem of sum of exponential variables to standard normal variable problem.

We perform 4 estimators of $\theta$.

1. we direct sample from Normal distribution and use indicator function to specifies whether larger than 0.89

$$\hat{\theta}_1 = \frac{1}{n}\sum_{i=1}^{n} I(X_i \geq 0.89), \text{ where } X_i \sim N(0,1)$$

2. Base on method 1, we use two side of sample to reduce the variance half

$$\hat{\theta}_2 = \frac{1}{2n}\sum_{i=1}^{n} I(|X_i| \geq 0.89), \text{ where } X_i \sim N(0,1)$$

3. Transform the normal variable to uniform variable

$$1 - 2\theta = \int_{-0.89}^{0.89} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx = 2\int_{0}^{0.89} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$$

$$2\theta = 1 - 2 \cdot 0.89 \int_{0}^{0.89} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \frac{1}{0.89} dx, \text{ where } X \sim U(0, 0.89)$$

$$\hat{\theta}_3 = \frac{1}{2} - \frac{0.89}{n}\sum_{i=1}^{n} \frac{1}{\sqrt{2\pi}} e^{-\frac{X_i^2}{2}}, \text{ where } X_i \sim U(0, 0.89)$$

4. Base on method 3, we transform $X$ to $\frac{1}{Y}$. So

$$\theta = \int_{-\infty}^{-0.89} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx = \frac{1}{0.89}\int_{-\frac{1}{0.89}}^{0} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2y^2}} \left|\frac{1}{y^2}\right| 0.89 dy, \text{ where } Y \sim U\left(\frac{1}{0.89}, 0\right)$$

$$\hat{\theta}_4 = \frac{1}{0.89n}\sum_{i=1}^{n} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2Y_i^2}}, \text{ where } Y_i \sim U\left(\frac{1}{0.89}, 0\right)$$

**Mean of Estimate** 下表是各種估計方法的估計值平均數，四種方法的估計值平均數都相當接近。

表 8: Mean of Estimates

| | method 1 | method 2 | method 3 | method 4 |
|---|---|---|---|---|
| N=100 | 0.1870 | 0.1860 | 0.1867 | 0.1865 |
| N=1000 | 0.1870 | 0.1870 | 0.1867 | 0.1870 |
| N=5000 | 0.1867 | 0.1867 | 0.1868 | 0.1867 |

**Standard Deviation of Estimate** 隨著樣本數提高，估計值的標準差都下降非常多。而method 3是標準差最低的方法，因為將normal variable轉換成uniform variable。method 4也是使用uniform variable，不過變數是method 3中的x取倒數，由於$\frac{1}{0.89} > 0.89$，也就是method 4中的隨機變數是來自Unif $\left(0, \frac{1}{0.89}\right)$，所以標準差會比method 3大。

表 9: Standard Deviation of Estimates

| | method 1 | method 2 | method 3 | method 4 |
|---|---|---|---|---|
| N=100 | 0.0388 | 0.0242 | 0.0035 | 0.0134 |
| N=1000 | 0.0123 | 0.0079 | 0.0011 | 0.0042 |
| N=5000 | 0.0055 | 0.0034 | 0.0005 | 0.0020 |

## Appendix

### R code

#### question 1

```
Hyperbolic <- function(q_x) {
  (-q_x^2/((1 - q_x)*log(1 - q_x)) - 0.04)^2}

nlminb(start = 0.04, objective = Hyperbolic)
UniDenth <- function(q_x) {
  (q_x/(1 - (1/2)*q_x) - 0.04)^2}

nlminb(start = 0.04, objective = UniDenth)
ConForce <- function(q_x) {
  (-log(1 - q_x) - 0.04)^2}

nlminb(start = 0.04, objective = ConForce)
```

#### question 2

```
y3s2 <- read_ods("y3s2-00000.ods", skip = 3)

colnames(y3s2)[1:4] <- c("Y", "year", "gender", "total")
male <- unique(y3s2$gender)[3]
```

```r
y3s2 <- y3s2[y3s2$gender %in% c(male) & y3s2$year %in% c(2019:2021),
             -c(1,3,4)]
colnames(y3s2) <- c("year", "0", "2", "7",
                    "12", "17" , "22", "27", "32", "37",
                    "42", "47" , "52", "57", "62", "67",
                    "72", "77" , "82", "87", "92", "97", "100")

y3s2_p <- transform(y3s2, `67` = as.numeric(`67`),
                    `100` = as.numeric(`100`))
kable(y3s2_p, digits = 1,
      caption = "Age Mortgage Rate in 2019-2021")

mr <- melt(y3s2, id = "year",
           variable.name = "age", value.name = "rate")
mr$age <- as.numeric(as.character(mr$age))
mr$rate <- as.numeric(mr$rate)
mr <- mr[mr$age != 0, ]
mr <- data.frame(x = mr$age, y = mr$rate/1000)
md4 <- nls(y ~ b * c^x, data = mr,
           start = list(b = 0.1, c = 1))
plot(mr$x, log(mr$y),
     main = "Mortgage Rate and Gompertz Model with Newton Method",
     xlab = "age", ylab = "log(rate)")
lines(mr$x, log(predict(md4)))
bc0 <- c(0.1, 1)
SSE <- function(y, x, bc){
  return(sum((y - (bc[1] * bc[2]^x))^2))}
estim <- marqLevAlg(bc0,
                    fn = SSE, x = as.matrix(mr$x), y = mr$y)

pred.y <- log(estim$b[1] * estim$b[2]^mr$x)
plot(mr$x, log(mr$y),
     main = "Mortgage Rate and Gompertz Model with LM method",
     xlab = "age", ylab = "log(rate)")
lines(mr$x, pred.y)
SSExy <- function(bc) {
  y <- mr$y
  x <- mr$x
  return(sum((y - (bc[1] * bc[2]^x))^2))
```

```
}
bc0 <- c(0.0001, 1.1)
stpd <- steep_descent(bc0, SSExy, maxiter = 1000)

pred.y <- log(stpd$xmin[1] * stpd$xmin[2]^mr$x)
plot(mr$x, log(mr$y),
     main = "Mortgage Rate and Gompertz Model with Steepest Descent",
     xlab = "age", ylab = "log(rate)")
lines(mr$x, pred.y)
summary_est <- rbind(
  Newton = c(summary(md4)[["coefficients"]][,1],
             summary(md4)[["sigma"]], md4$convInfo[["finIter"]]),
  LM = c(estim$b, estim$fn.value, estim$ni),
  SteepDesc = c(stpd$xmin, stpd$fmin, stpd$niter)
)
colnames(summary_est) <- c("B.est", "C.est", "SSE", "iter")
kable(summary_est, digits = 5,
      caption = "Summary of three Methods of Optimization")
```

**question 3**

```
Likelihood <- function(theta) {
  estimator = (1997/(2 + theta)) - (1810/(1 - theta)) + 32/theta
  return(estimator)}

DerivLikeli <- function(theta) {
  estimator = (-1997/(2 + theta)^2) + (1810/(1 - theta)^2) - 32/theta^2
  return(estimator)}

NewtonMethod <- function(estimator, threshold) {
  e <- 1
  iter <- 0
  while(e > threshold) {
    y = estimator - Likelihood(estimator)/DerivLikeli(estimator)
    e = abs(y - estimator)
    estimator = y
    iter = iter + 1}
  return(data.frame(ans = y, Likelihood = Likelihood(y),
```

```
                    iteration = iter))}

NewtonMethod(0.05, 10^-6)
SecantMethod <- function(a, b, threshold) {
  if(Likelihood(a)*Likelihood(b) > 0) {
    return('No solution by SecantMethod.')}

  else{
    iter <- 0
    e <- 1
    while(e > threshold) {
      new = b - Likelihood(b)*(b - a)/(Likelihood(b) - Likelihood(a))
      a = b
      b = new
      e = abs(Likelihood(new))
      iter = iter + 1}}
  return(data.frame(ans = new, Likelihood = Likelihood(new),
                    iteration = iter))}

SecantMethod(0.01, 0.09, 10^-6)
RidderMethod = function(a, b, threshold) {
  if(Likelihood(a)*Likelihood(b) > 0) {
    return('No solution by RidderMethod.')}

  else{
    iter <- 0
    e <- 1
    while(e > threshold) {
      c = (a + b)/2
      new = c + ((c - a)*(sign(Likelihood(a) - Likelihood(b))*Likelihood(c))
                 / sqrt(Likelihood(c)^2 - Likelihood(a)*Likelihood(b)))

      if(Likelihood(new) < 0) {
        a = a
        b = new}

      else{
        a = new
        b = b}
```

```
      e = abs(Likelihood(new))
      iter = iter + 1
    }}
  return(data.frame(ans = new, Likelihood = Likelihood(new),
                    iteration = iter))}

RidderMethod(0.01, 0.09, 10^-6)
```

**question 4**

```
ImportantSample <- function(x, n){
  u <- runif(n)
  y <- - (sqrt(3)/pi) * log((1 + exp(-pi*x/sqrt(3))) / u - 1)
  gy <- ((pi * exp(-pi*y/sqrt(3))) /
           (sqrt(3) * (1 + exp(-pi*y/sqrt(3)))^2))
  phiy <- dnorm(y)
  est <- ((1/n) / (1 + exp(-pi*x/sqrt(3)))) * sum(phiy / gy)
  return(est)
}

MCInteg <- function(x, n){
  u <- runif(n, 1/x, 0)
  y <- (1 / (sqrt(2*pi))) * exp(-1/(2*u^2)) / (abs(x)*u^2)
  est <- sum(y) / n
  return(est)
}

AntMCInteg <- function(x, n){
  u <- runif(n, 1/x, 0)
  x_u <- 1/x - u
  u2 <- c(u, x_u)
  y <- (1 / (sqrt(2*pi))) * exp(-1/(2*u2^2)) / (abs(x)*u2^2)
  est <- sum(y) / (2*n)
  return(est)
}

x <- c(-6, -5, -4, -3.5, -3, -2.5, -2)
n <- c(100, 1000, 5000)
```

```
res_is <- outer(x, n, Vectorize(function(x, n){
  mean(replicate(1000, ImportantSample(x = x, n = n)))}))
res_is <- cbind(res_is, pnorm(x))
dimnames(res_is) <- list(paste0("x=", x),
                         c(paste0("n=", n), "pnorm"))
kable(res_is, digits = 10,
      caption = "Mean of Estimates with Important Sampling")
res_mci <- outer(x, n, Vectorize(function(x, n){
  mean(replicate(1000, MCInteg(x = x, n = n)))}))
res_mci <- cbind(res_mci, pnorm(x))
dimnames(res_mci) <- list(paste0("x=", x),
                          c(paste0("n=", n), "pnorm"))
kable(res_mci, digits = 10,
      caption = "Mean of Estimates with Monte-Carlo Integration")
res_amci <- outer(x, n, Vectorize(function(x, n){
  mean(replicate(1000, AntMCInteg(x = x, n = n)))}))
res_amci <- cbind(res_amci, pnorm(x))
dimnames(res_amci) <- list(paste0("x=", x),
                           c(paste0("n=", n), "pnorm"))
kable(res_amci, digits = 10,
      caption = "Mean of Estimates with Antithetic Monte-Carlo Integration")
n <- 100
sd_is <- outer(x, n, Vectorize(function(x, n){
  sd(replicate(1000, ImportantSample(x = x, n = n)))}))
sd_mci <- outer(x, n, Vectorize(function(x, n){
  sd(replicate(1000, MCInteg(x = x, n = n)))}))
sd_amci <- outer(x, n, Vectorize(function(x, n){
  sd(replicate(1000, AntMCInteg(x = x, n = n)))}))

summary_sd <- cbind(Important = sd_is,
      MCInteg = sd_mci,
      AntiMCInteg = sd_amci)

colnames(summary_sd) <- c("ImportantSample", "MCInteg", "AntMCInteg")
row.names(summary_sd) <- paste0("x=", x)
kable(summary_sd, digits = 10,
      caption = "Standard Deviation of Estimates with n=100")
n <- 5000
sd_is <- outer(x, n, Vectorize(function(x, n){
```

```
  sd(replicate(1000, ImportantSample(x = x, n = n)))}))
sd_mci <- outer(x, n, Vectorize(function(x, n){
  sd(replicate(1000, MCInteg(x = x, n = n)))}))
sd_amci <- outer(x, n, Vectorize(function(x, n){
  sd(replicate(1000, AntMCInteg(x = x, n = n)))}))

summary_sd <- cbind(Important = sd_is,
      MCInteg = sd_mci,
      AntiMCInteg = sd_amci)

colnames(summary_sd) <- c("ImportantSample", "MCInteg", "AntMCInteg")
row.names(summary_sd) <- paste0("x=", x)
kable(summary_sd, digits = 10, caption = "Standard Deviation of Estimates with n=5000")
```

**question 5**

```
###. Real Value
funct = function(x) exp(x^2)
value = round(c(integrate(funct, 0, 1))$value, 6)

###. Numerical Integration
start = 0
NumInt = function(n) {
  gap = 1/n
  start = gap/2
  for (i in 1:(n - 1)) {
    start = c(start, i*gap + gap/2)
  }
  return(sum(funct(start))/n)}

###. Monte Carlo Integration
MonteCar = function(n) {
  random = runif(n)
  random_value = funct(random)
  estimate = mean(random_value)
  standerr = sd(random_value)/sqrt(n)
  err = abs(estimate - value)
  r1 = round(c(estimate, standerr, err), 6)
```

```
  return(r1)}

###. Antithetic Method
AntiMeth = function(n) {
  random = runif(n)
  random_anti = c(random, 1 - random)
  random_value = funct(random_anti)
  estimate = round(mean(random_value), 6)
  standerr = sd(funct(random) + funct(1 - random))/sqrt(2*n)
  err = abs(estimate - value)
  r2 = round(c(estimate, standerr, err), 6)
  return(r2)}

###. Stratified Sampling Method
StratSamp = function(n, m, N) {
  L = seq(0, 1, length = m + 1)
  Vtheta_S <- matrix(0, N, 2)
  for(i in 1:N) {
    random_value <- funct(runif(n))
    Vtheta_S[i,1] <- mean(random_value)
    theta_MCJ <- c()
    for (j in 1:m) {
      theta_MCJ[j] <- mean(funct(runif(n/m, L[j], L[j+1])))}
    Vtheta_S[i,2] <- mean(theta_MCJ)}
  estimate = round(apply(Vtheta_S, 2, mean)[2], 6)
  standerr = apply(Vtheta_S, 2, sd)[2]
  err = abs(estimate - value)
  r3 = round(c(estimate, standerr, err), 6)
  return(r3)}

Com = function(n) {
  result = cbind(c(value, "", ""), c(NumInt(n), "", abs(NumInt(n) - value)),
                                    MonteCar(n), AntiMeth(n/2),
                                    StratSamp(n, 4, 1000))
  colnames(result) = c("Real Value", "Numerical ", "Monte Carlo ",
                       "Antithetic ", "Stratified Sampling ")
  rownames(result) = c("Estimate", "SE", "Error")
  return(result)}
```

```r
set.seed(SEED)
list('n = 50' = Com(50), 'n = 100'= Com(100),
     'n = 500'= Com(500),'n = 1000' = Com(1000))
```

**question 6**

```r
lambda <- 1
num <- 5
critical <- 21.6

mean_s <- sum(seq(1, num))
var_s <- sum((seq(1, num))^2)
sd_s <- sqrt(var_s)
std_c <- (critical - mean_s) / sd_s

# 4 methods to generate estimate of theta
Rtheta1 <- function(n, std_c){
  sum(rnorm(n) > std_c) / n
}

Rtheta2 <- function(n, std_c){
  sum(abs(rnorm(n)) > std_c) / (2*n)
}

Rtheta3 <- function(n, std_c){
  u <- runif(n, min = 0, max = std_c)
  0.5*(1 - sum(2 * std_c * (1 / sqrt(2*pi)) * exp(-(u^2) / 2)) / n)
}

Rtheta4 <- function(n, std_c){
  y <- runif(n, min = 0, max = 1/std_c)
  sum((1 / std_c) * (1 / sqrt(2*pi)) * exp(-1 / (2*y^2)) / (y^2)) / n
}

# function that perform Rtheta simulation 1000 times
EstTheta <- function(Rtheta, n, std_c, time = 1000, var = TRUE){
  ests <- replicate(time, Rtheta(n, std_c))
  if (var == TRUE){
```

```
    return(sd(ests))
  }else{
    return(mean(ests))
  }
}

Rtheta_list <- c(rtheta1=Rtheta1, rtheta2=Rtheta2,
                 rtheta3=Rtheta3, rtheta4=Rtheta4)
n <- c(100, 1000, 5000)
mean_est <- outer(n, Rtheta_list,
      Vectorize(function(n, Rtheta){
        EstTheta(Rtheta, n = n, std_c = std_c, var = FALSE)}))
dimnames(mean_est) <- list(paste0("N=", n),
                           paste("method", 1:4))
kable(mean_est, digits = 4, caption = "Mean of Estimates")
sd_est <- outer(n, Rtheta_list,
                Vectorize(function(n, Rtheta){
                  EstTheta(Rtheta, n = n, std_c = std_c)}))
dimnames(sd_est) <- list(paste0("N=", n), paste("method", 1:4))
kable(sd_est, digits = 4, caption = "Standard Deviation of Estimates")
```