Statistical Computing and Simulation HW3

授課教授：余清祥教授

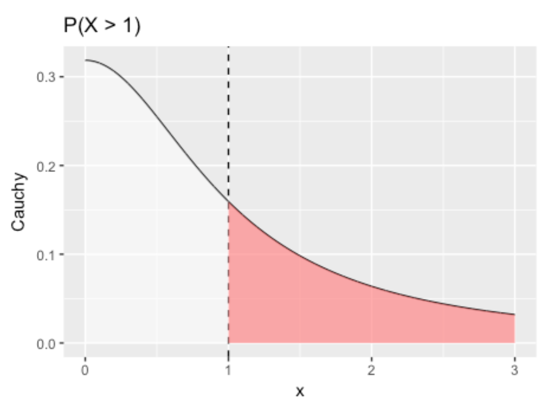
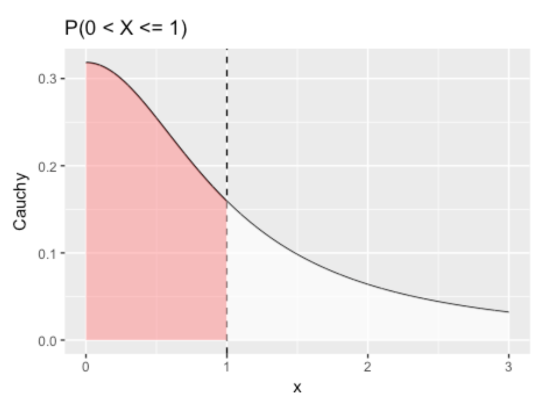
學生: 統計碩一 106354003 林健宏

統計碩一 106354012 曹立諭

目錄

Question 01.

Experiment with as many variance reduction techniques as you can think of to apply the problem of evaluating $P(X > 1)$ for $X \sim$ Cauchy.

(圖一)	(圖二)
	

以 Monte-Carlo Integration 為基準，並使用其他變異數縮減方法去降低估計的誤差，使用方法有以下幾種：Hit or miss、Antithetic Variate、Importance Sampling、Control variate 以及 Stratified Sampling。每一個方法皆取 1000 個隨機樣本生成一個$\hat{\theta}$，再生成 1000 樣本 $\hat{\theta}$，並運用其平均去估計真實的θ及計算樣本變異數。

使用方法介紹

(a) Monte-Carlo Integration：

$P(0 < X \leq 1)$可以經由 Monte-Carlo Integration 模擬，x 從 uniform(0,1)抽取 1000 筆資料，並將每筆資料帶入 $f(x) = \frac{1}{\pi(1+x^2)}$ 得到每筆資料的對應值，再取這 1000 筆資料對應值的平均值為 $P(0 \leq X \leq 1)$的估計值。因此 $P(X > 1)$的估計值為 0.5 - $P(0 < X \leq 1)$的估計值。我們重複這動作 1000 次取得 1000 個 $P(X > 1)$的估計值，計算這 1000 個 $P(X > 1)$的估計值的平均與變異數為 0.24997910 與 0.00000272，此為 Monte-Carlo Integration。

(b) Hit or miss

從 $X \sim$ Cauchy 中隨機抽取 1000 筆資料，計算幾筆資料超過 1 的比例即為$\theta = P(X > 1)$的估計值，重複這個動作直到取得 1000 筆 $\theta = P(X > 1)$的估計值，最後取這 1000 筆估計值的平均與變異數，分別為 0.2498500 與 0.000193，此為 Hit or miss 估計方法。

(c) Antithetic Variate

做法類似 Monte-Carlo Integration，x 從 Uniform(0,1)抽取 500 筆，再將這 500 筆資料用 $1-x$ 的方式生成另外 500 筆資料，形成 1000 筆資料。後續步驟雷同 Monte-Carlo Integration 的步驟，得到 $P(X>1)$的估計值的平均與變異數為 0.24999466 與 0.00000004，此為 Antithetic Variate 縮小變異數方法。

(d) Importance Sampling

先選找另一個函數 g(x)（此題使用的為$\frac{1}{x^2}$），令 $Y=\frac{g(x)}{f(x)}$，$\int g(x)dx=\int \frac{g(x)}{f(x)} f(x)dx=E(Y)$，再利用 Monte Carlo 方法去估計 E(Y)。當 Y 越接近常數，變異數縮減效果越好。因此我們透過此方法，形成 1000 筆 $P(X>1)$的估計值，並取這 1000 筆 $P(X>1)$的估計值的平均數與變異數為 0.24997706 與 0.00000244，

此為 Importance Sampling 縮小變異數方法。

(e) Control variate

先找一個與本題函數$f(x)=\frac{1}{\pi(1+x^2)}$類似的函數（此題使用 $g(x)=\frac{1}{1+x}$），這個函數要能求取期望值，利用 $\widehat{\theta_a}=f(x)+a(g(x)-E(g(x)))$，$a=-\frac{Cov(f(x),g(x))}{Var(g(x))}$, x 從 uniform(0,1)隨機抽取 1000 筆，將這 1000 筆 x 代入 $\widehat{\theta_a}$並取平均即為 $P(0<X\leq 1)$的估計值，0.5 - $\widehat{\theta_a}$ 為 $P(X>1)$的估計值。總共取 1000 筆 $P(X>1)$的估計值，並取並取這 1000 筆 $P(X>1)$的估計值的平均數與變異數為 0.24982097 與 0.00000983，此為 Control variate 縮小變異數方法。

(f) Stratified Sampling

做法類似於 Monte-Carlo Integration，我們將 x 的範圍(0,1)切成 5 等分，即(0, 0.2), (0.2, 0.4), …, (0.8,1)，每做一次 $P(0\leq X\leq 1)$的估計值，要抽取 1000 個 x 樣本，這邊將 x 範圍等分成 5 等份，因此樣本也平均分配成每一區間各抽取 200 個樣本並帶入$f(x)=\frac{1}{\pi(1+x^2)}$，目的是為了讓樣本的分散程度足夠、代表性足夠，將這 5 組資料分別取平均後得到 $\widehat{\theta_{11}}, \widehat{\theta_{12}}, \ldots, \widehat{\theta_{15}}$，再將這 5 個估計值取平均即為$P(0<X\leq 1)$的估計值$\widehat{\theta_1}$, 而$P(X>1)$的估計值為 0.5 - $\widehat{\theta_1}$。重複上述動作 1000 次，最後取這 1000 筆估計值的平均與變異數，分別為 0.25001367 與 0.00000009，此為 Stratified Sampling 估計方法。

$$\text{實際}\theta \approx P(X > 1) = 0.25$$

以 Monte-Carlo Integration 為基準，並使用其他變異數縮減方法去降低估計的誤差，使用方法有以下幾種：Hit or miss、Antithetic Variate、Importance Sampling、Control variate 以及 Stratified Sampling

（其中用於 Control variate 的另一函數為$\frac{1}{1+x}$, x = 0 ~ 1，Stratified Sampling 將定義域分為 5 層並平均分配抽樣數，用於 Importance Sampling 的另一函數為$\frac{1}{x^2}$ $, x = 0 \sim 1$。）

每一個方法皆取 1000 個隨機樣本生成一個$\hat{\theta}$，再生成 1000 樣本 $\hat{\theta}$，並運用其平均去估計真實的 θ及計算樣本變異數。

統整六種方法估計 P(X > 1)的估計值樣本平均數與樣本變異數：

	MonteCarlo	Hit or miss	Antithetic	Importance	Control	Stratified(5)
$E(\hat{\theta})$	0.24997910	0.2498500	0.24999466	0.24997706	0.25017903	0.25001367
$var(\hat{\theta})$	0.00000272	0.0001932	0.00000004	0.00000244	0.00000983	0.00000009

另外亦有探討抽樣樣本數的問題，分別取 100、1000、10000 樣本數，去探討樣本平均以及樣本變異數之間的關係。

在此以表格方式呈現

n = 100	Monte-Carlo	Hit or miss	Antithetic	Importance	Control	Stratified(5)
$E(\hat{\theta})$	0.25028315	0.24937000	0.25002894	0.24989915	0.24878815	0.2500383
$var(\hat{\theta})$	0.00002736	0.00185055	0.00000041	0.00002551	0.00009001	0.0000008

n = 1000	Monte-Carlo	Hit or miss	Antithetic	Importance	Control	Stratified(5)
$E(\hat{\theta})$	0.24997910	0.2498500	0.24999466	0.24997706	0.25017903	0.25001367
$var(\hat{\theta})$	0.00000272	0.0001932	0.00000004	0.00000244	0.00000983	0.00000009

n = 10000	Monte-Carlo	Hit or miss	Antithetic	Importance	Control	Stratified(5)
$E(\hat{\theta})$	0.25001537	0.24978600	0.2499985	0.24997728	0.25001918	0.25000063
$var(\hat{\theta})$	0.00000025	0.00001847	0.0000001	0.00000024	0.00000099	0.00000001

(g) 結論：

實際$\theta = P(X > 1) = 0.25$，由上表可知，每個方法都估計的數值相差無幾，只有 Antithetic Variate 與 Stratified Sampling 的樣本變異數來得非常小，而 Control variate 與 Hit or miss 來得相對大滿多，其中 Control variate 可能與取得函數有關，若是能取得更適合的函數，也許可以將樣本變異數降低更多。

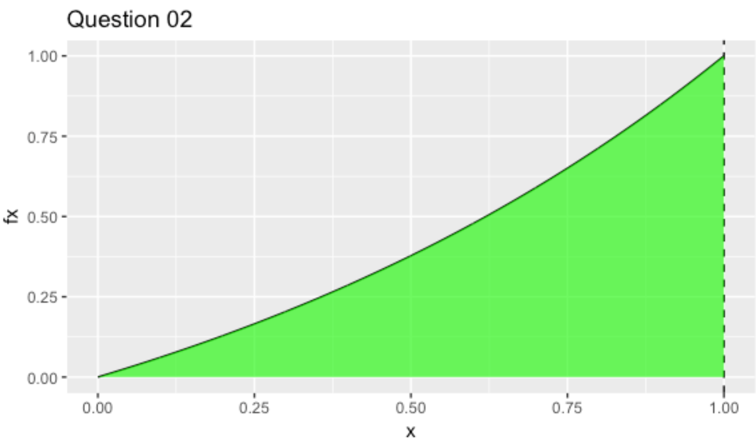
而抽樣樣本數也是一個問題，當樣本數由小至大會發現樣本數的樣本變異數會下降很快，會近乎接近0。因此，在本題使用的估計方法以Antithetic Variate與Stratified Sampling為優，樣本數則是越多誤差越小。

Question 02.

Hammersley and Handscomb (1964) used the integration of $\theta = \int_0^1 \frac{e^x-1}{e-1}\,dx$ on $(0,1)$

as a test problem of variance reduction techniques (which is about 0.4180233).
Achieve as large a variance reduction as you can. (They achieved 4 million.)

實際 $\theta = \int_0^1 \frac{e^x-1}{e-1}\,dx \approx 0.4180233$



以 Monte-Carlo Integration 為基準，並使用其他變異數縮減方法降低估計的誤差，使用方法有以下幾種： Antithetic Variate、Importance Sampling、Control variate 以及 Stratified Sampling。每一個方法皆取 1000 個隨機樣本生成一個$\hat{\theta}$，再生成 1000 樣本 $\hat{\theta}$，並運用其平均去估計真實的θ及計算樣本變異數。

其中本題運用方法方式與第一題類似，而用於 Control variate 的另一函數為x ,$0 < x < 1$，Stratified Sampling 將 x 的範圍 0~1 平均分為 5 層並平均分配抽樣數，

用於 Importance Sampling 的另一函數為 $g(x) = \frac{4}{\pi(1+x^2)}$ ，$0 < x < \infty$。

另外本題亦有多做縮減變異數混搭，在 Importance Sampling 的方法之中，
套入 Antithetic Variate 抽取 x 樣本的方法以藉此來縮小只做 Importance Sampling 的變異數。

統整六種方法估計 θ =$\int_0^1 \frac{e^x-1}{e-1}dx$估計值的樣本平均數與樣本變異數：

	Monte-Carlo	Antithetic	Importance	Control	Stratified(5)
$E(\hat{\theta})$	0.41787975	0.41805356	0.41824956	0.4180233	0.41804309
$var(\hat{\theta})$	0.00008453	0.00000281	0.00015477	9.253717e-36	0.00000345

結論：

實際 $\theta = \int_0^1 \frac{e^x-1}{e-1}dx \approx 0.4180233$，由上表可知，每個方法估計 θ =$\int_0^1 \frac{e^x-1}{e-1}dx$的數值相差無幾，多落在 0.417~0.418 附近。

接著比較每個方法變異數，以 Antithetic Variate、Stratified Sampling 與 Control variate 的變異數最小，而 Importance Sampling 的變異數則是六個之中最大的，可能與 Importance Sampling 使用的另一函數（$\frac{4}{\pi(1+x^2)}$）有關，若是能取得更適合的函數，便可以將樣本變異數降下來。

在本題中，我們亦有做縮減變異數方法的混搭，透過兩個方法的結合（Importance Sampling 與 Antithetic Variate），來讓 Importance Sampling 方法中的變異數進一步縮減，從 0.00015477 縮減至 0.00004137。因此本題以 Antithetic Variate、Stratified Sampling 與 Control variate 三個方法為優，另外能透夠增加抽樣樣本數來降低所有方法之變異數。

Question 03.

Let $X_i, i = 1,2,3,4,5$ be independent exponential random variables each with mean 1,
and consider the quantity θ defined by $\theta = P(\sum_{i=1}^{5} iX_i \geq 21.6)$.
Propose at least three simulation methods to estimate θ and compare their variances.

以 Monte-Carlo Integration 為基準，並使用其他變異數縮減方法去降低估計的誤差，使用方法有以下幾種：Hit or miss、Antithetic Variate 以及 Stratified Sampling。在使用 Monte-Carlo Integration、Antithetic Variate 以及 Stratified Sampling 時，需要$\sum_{i=1}^{5} iX_i$的機率分配形式，透過變數變換後，

$$y = \sum_{i=1}^{5} iX_i \text{的機率分配：} \frac{312.5}{60}e^{-\frac{y}{5}} - \frac{640}{60}e^{-\frac{y}{4}} + \frac{405}{60}e^{-\frac{y}{3}} - \frac{4}{3}e^{-\frac{y}{2}} + \frac{1}{24}e^{-y}\text{，} y \geq 0$$

先計算$P(y < 21.6)$的估計值，再透過$1 - P(y < 21.6) = P(\ y \geq 21.6)$的方式取得估計值。

以 Monte-Carlo Integration 為例，先從 uniform(0,21.6)抽取 1000 筆樣本，帶入 y 的機率形式中並乘上 21.6 後去取平均數即為一個$P(y < 21.6)$的估計值，再計算$1 - P(y < 21.6)$的估計值即為$P(\ y \geq 21.6)$的估計值。

重復此動作 1000 次，取得 1000 筆$P(y \geq 21.6)$的估計值後，取此 1000 筆估計值的平均數與變異數，即為 0.16960651 與 0.00018478。
而 Antithetic Variate 與 Stratified Sampling 做法類似 Question 01 的做法，便能取得該方法所估計$P(\ y \geq 21.6)$估計值的平均數與變異數。

Hit or miss 方法則是直接從$X \sim \exp(1)$分配中抽取 5 筆資料，並計算$\sum_{i=1}^{5} iX_i$的數值，
重複取 1000 筆$\sum_{i=1}^{5} iX_i$並計算超過 21.6 的比例為一個$P\left(\sum_{i=1}^{5} iX_i \geq 21.6\right)$的估計值。
重復抽取 1000 筆$P\left(\sum_{i=1}^{5} iX_i \geq 21.6\right)$的估計值，取其平均數與變異數為 0.16906200 與 0.00015192。

在此以表方式統整四種方法估計 $\theta = P\left(\sum_{i=1}^{5} iX_i \geq 21.6\right)$估計值的樣本平均數與樣本變異數：

	Monte-Carlo	Antithetic	Stratified(5)	Hit or miss
$E(\hat{\theta})$	0.16810602	0.16730692	0.16894173	0.16816900
$var(\hat{\theta})$	0.00019783	0.00029715	0.00002004	0.00013109

結論：

在本題四種方法中，估計的 $\hat{\theta}$ 大多都落在 0.168 附近，唯獨 Antithetic Variate 與 0.168 有一點落差，變異數的部分則是 Stratified Sampling 最小，Antithetic Variate 最大。
因此，本題以 Stratified Sampling 為優。

Question 04.

First, simulate 100 observations from $\text{Beta}(2,3)$ and then use 3 density estimating methods to smooth the observations. You need to specify the parameters in the smoothing methods, and compare the results.

我們使用了下列三個方法來估計我們的密度函數Beta(2,3)，分別是

(a) Histogram

我們將亂數取出來 min(x)、max(x) 作為我們兩端的範圍，將此範圍切割成適當的大小 m (依照亂數的個數決定)，$\mathrm{a} = \mathrm{a}_0 < a_1 < \cdots < \mathrm{a}_{\mathrm{m}-1} < a_m = \mathrm{b}$，
此密度函數的估計值為

$$\hat{f}(x) = \frac{1}{n}\sum_{j=1}^{m}\frac{n_j}{h} \times I\{x \in [a_{j-1}, a_j]\} \quad \mathrm{n_j} \text{ 為 } [a_{j-1}, a_j] \text{ 裡的個數}$$

其中 h 為環寬，寬度會影響整個估計函數矩形的形狀及大小

(b) The Naïve Density Estimator

我們將亂數取出來 min(x)、max(x) 作為我們兩端的範圍，將此範圍切割成適當的大小 m (依照亂數的個數決定)，$\mathrm{a} = \mathrm{a}_0 < \mathrm{a}_1 < \cdots < \mathrm{a}_{\mathrm{m}-1} < \mathrm{a}_{\mathrm{m}} = \mathrm{b}$，
此密度函數的估計值為

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h}w\left(\frac{x-x_i}{h}\right) \text{，} w\left(\frac{x-x_i}{h}\right) = \frac{1}{2} \text{ ,when } \left|\frac{x-x_i}{h}\right| < 1$$

其中 h 為環寬，寬度會影響整個估計函數曲線的起伏程度

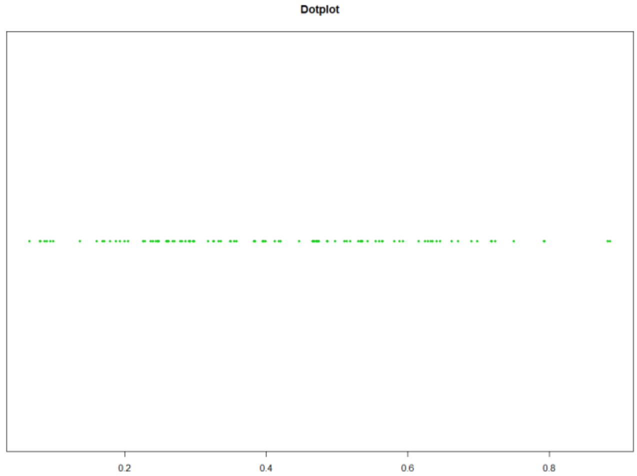
(c) Kernel Estimator

我們將亂數取出來 min(x)、max(x) 作為我們兩端的範圍，將此範圍切割成適當的大小 m (依照亂數的個數決定)，$\mathrm{a} = \mathrm{a}_0 < \mathrm{a}_1 < \cdots < \mathrm{a}_{\mathrm{m}-1} < \mathrm{a}_{\mathrm{m}} = \mathrm{b}$，
此密度函數的估計值為

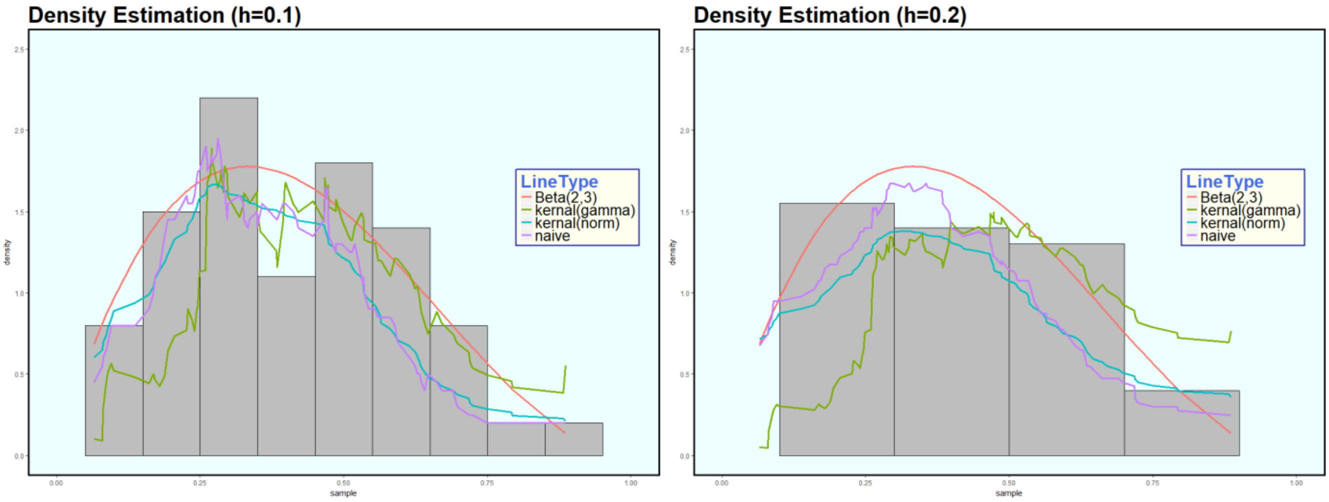
$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h}K\left(\frac{x-x_i}{h}\right) \text{，} \int_{-\infty}^{\infty} K(x)\,dx = 1$$

其中 h 為環寬，寬度會影響整個估計函數曲線的平滑程度，
K 為一機率密度函數，這裡我們使用標準常態分配作為核密度函數，
並用此算式來計算新的估計值，並將結果繪製成圖表。

這是我們模擬出的 x 散佈圖，接者將這先模擬出的亂數帶入上面提及之方法函數，將各方法所得出的新值繪製在相同圖表上比較，



我們使用上述三種方法來將模擬出的新數值進行繪製，並將圖形堆疊至相同圖表進行比較



（ 橘色線為 Beta(2,3) ； 綠色線為 kernel(Gamma) ； 藍綠色線為 kernel(norm) ； 紫色線為 naïve ）

從上述兩圖中我們可以發現不同 h 的選擇會讓曲線的平滑程度跟起伏程度有所改變，以此題分配模擬出的 100 個亂數中，我們發現 h=0.1 所繪製的圖表較為準確，而 h=0.2 所繪製的圖表其高度都有所下降，與前述的圖表比較有稍微平滑一點。

這裡我們選擇了兩種核密度函數來估計函數，從繪製的圖形中我們可以看出，使用 Gamma 函數來估計函數沒有比使用常態函數來估計的好，這裡是個之後可以再深入探討的點，該使用哪種核密度函數來估計會是比較好的方法，而在此題之中，我們一致認定使用常態分配的函數會有較好的準確程度

Question 05.

Let x be 100 equally spaced points on $[0,2\pi]$ and let $y_i = \sin x_i + \epsilon_i$ with $\epsilon_i \sim N(0,0.09)$. Apply at least 3 linear smoothers and compare the differences, with respect to mean squares error (i.e., bias^2 and variance) from 1,000 simulation runs.

這裡我們使用了下列四種方法使模擬點的曲線變得更平滑

(a) Kernel Smoothers

$$\hat{y}_i = \sum_{j \epsilon N_i} w_{ij} y_j \quad , w_{ij} = \frac{K(\frac{x_i - x_j}{h})}{\sum_{j \epsilon N_i} K(\frac{x_i - x_j}{h})}$$

根據上式函數，我們將模擬值帶入並計算出新的估計值，因為 K 為機率密度函數，當我們將新的估計值繪出線段時會較原本模擬值來的平滑許多。

(b) Lowess (局部加權迴歸)

我們將模擬值依序排列，可以設定每次使用多少個鄰近模擬值，配飾出該點的迴歸線，並計算該點新的模擬值，最後將其繪出成較平滑的線段。

(c) Spline smooth

首先將原始觀察值分成 k+1 個區間，其中$k1, k2, \ldots, k_k$為這 k+1 個區段內多項式的交接點，我們一般在每個區間內使用一個三次多項式，並利用模擬值來配飾出各區間的三次式，再依照各區間及三次式匯出成較平滑的線段。
（其線段會一節點選擇而有不同的平滑程度）

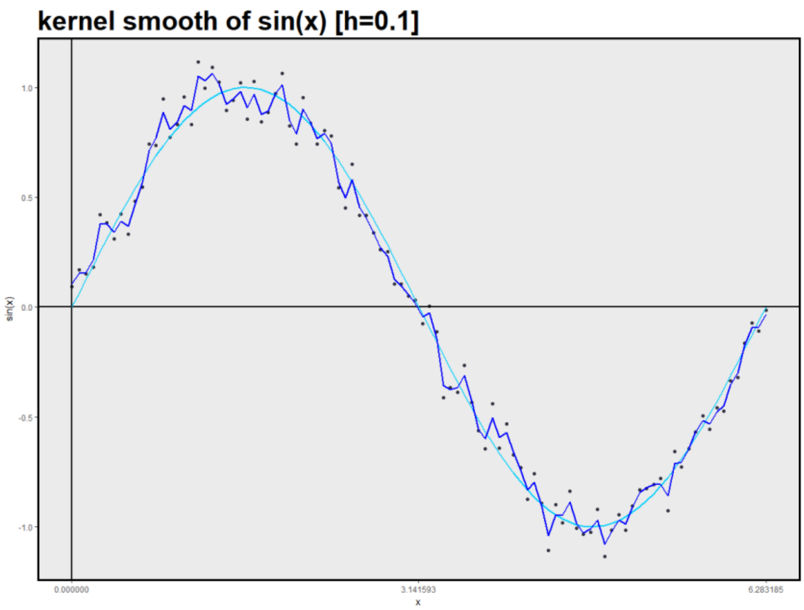
(d) Running means (移動平均法)

我們將模擬值依序列出，並設定我們想要的移動平均的個數，其新的模擬值為該點周圍 K 個模擬值的平均，最後將其新值繪出成線段。

當曲線使用這四種方法平滑後，我們分別模擬不同方法 1000 次，並計算各方法之均方誤差 MSE，其為預測誤差平方值總和的平均。

(a) Kernel smooth

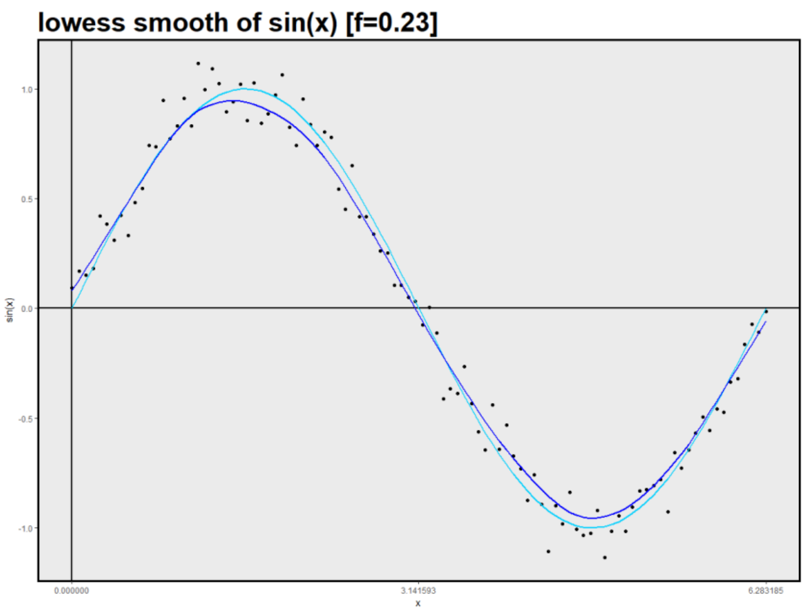
在這裡我們使用了核密度函數為標準常態分配的方法來繪製線段，可以看出線段有比模擬值點兩點間的連線來的稍微平滑，如果我們選取的核密度函數不同，則會有不同的平滑程度。



（淺藍色線段為 sin(x)；深藍色線段為 kernel smooth line）

(b) Lowess

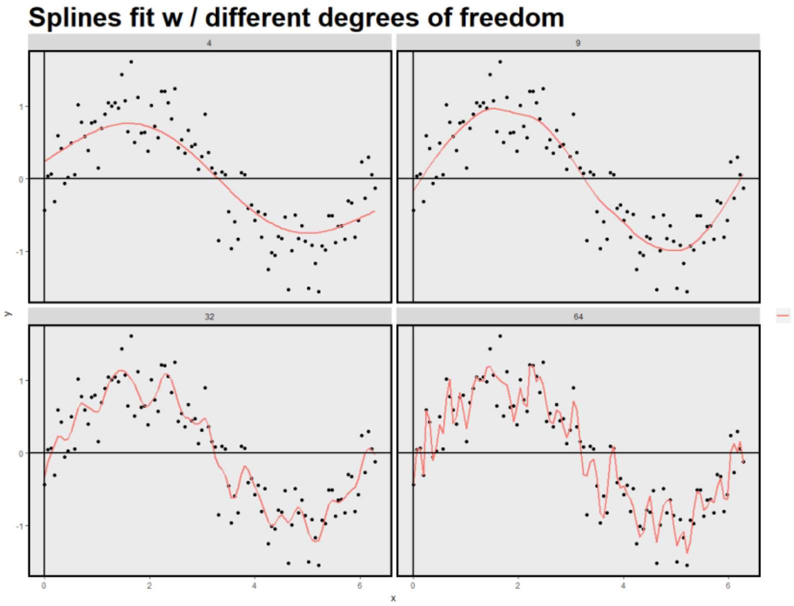
這裡我們先在 R 軟體設定 f 變數為 0.23，這給出了影響每個值平滑的圖中點的比例，而較大的值會使平滑度更高，我們找到一個適合我們原本圖形的平滑程度的比例，並將其繪製出線段。



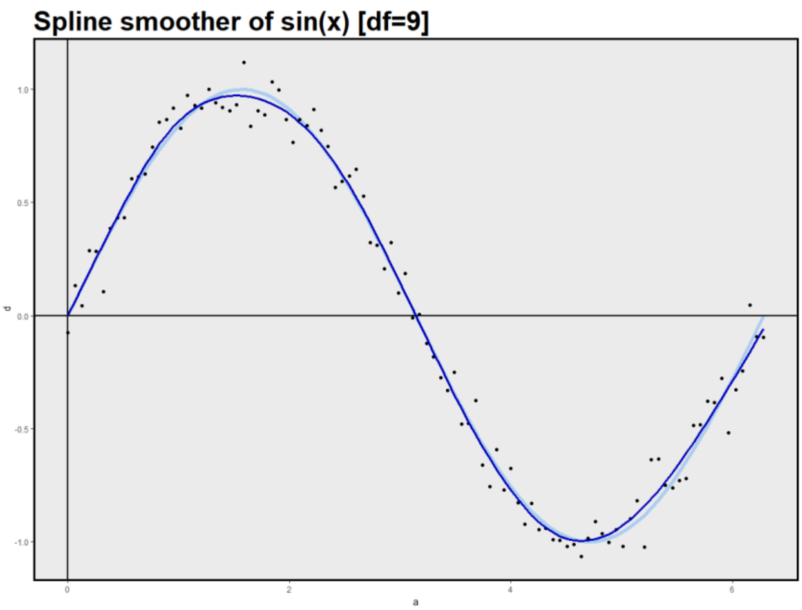
（淺藍色線段為 sin(x)；深藍色線段為 lowess smooth line）

(c) Spline smooth

我們先按照節點選取的數量分別畫出平滑線來觀看，並發現不同節點的選擇會讓平滑線的曲度不一樣，這裡我們選取了自由度分別為 4、9、32、64，共四組來畫出不同自由度下的平滑曲線。

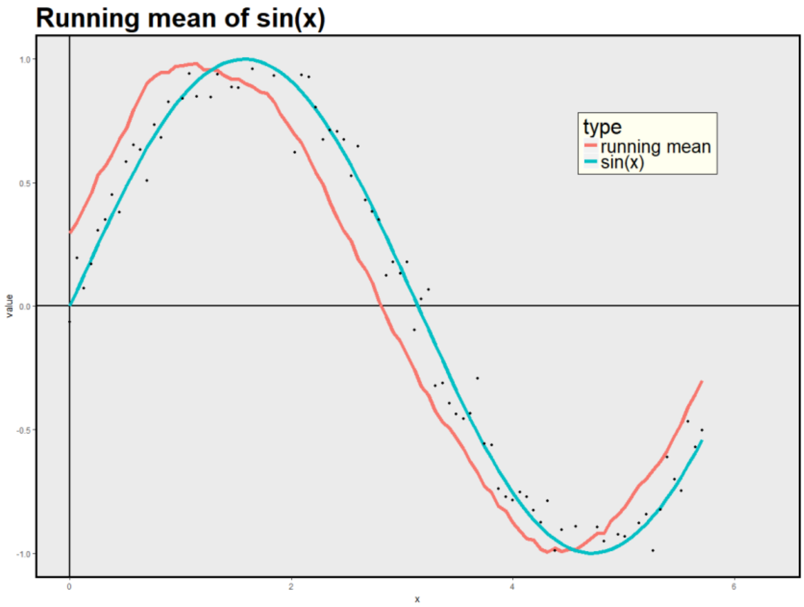


我們發現當自由度選擇過大時，平滑線的曲線反而會變得震盪許多，接著我們將各自由度帶入計算其 MSE，發現在模擬 1000 次後的結果中，自由度為 9 的平滑曲線其 MSE 最小，並將其自由度繪出圖形觀看，發現其曲線與真實的 sin 函數圖形幾乎重疊。



（淺藍色線段為 sin(x)；深藍色線段為 Spline smooth line）

(d) Running mean smooth



（淺藍色線段為 sin(x)；紅色線段為 Running mean smooth line）

(e)結論

我們使用這四種方法，模擬 1000 次並計算其 MSE，呈現在下列表格中

	Kernel smooth	Lowess smooth	Cubic Spline	Running Means
MSE($\hat{\theta}$)	0.004201074	0.00161815	0.0006245836	0.004054571

從上圖表格我們可以發現 Spline smoother 的 MSE 最小，而從我們上述所繪出的線段判斷來說，Spline smooth 的線段確實很靠近我們想要估計的 sin 函數線段。

我們覺得 Cubic Spline smooth 的方法最好，但如果當模擬值過多時，我們發現其運算會很耗費時間，因為當結點數越多時，其矩陣的大小就越大，會因為矩陣的大小影響運行速度。

❒ 附錄（R code）

Github : https://github.com/CaoCharles/Statistical-Computing-and-Simulation-HW3
R Markdown :
1.

```
# Cauchy為中心點為0的對稱分佈
# P(X>1)=0.25、P(0<X<1)=0.25
# 接著我們使用下列方法來降低變異數
# 都取1000個theta.hat去估計theta
# Monte-Carlo Integration -----------------------------------------------

N <- 1000
cauchy <- function(x){
  return(1/(pi*(1+x^2)))
}
theta.hat <- NULL
for(i in 1:1000){
  # 從cauchy抽取樣本
  x <- runif(N)
  x <- cauchy(x)
  theta.hat[i]<-mean(x)}
Mc.mean <- mean(theta.hat)
Mc.var <- var(theta.hat)
```

```
# Hit or miss1 -----------------------------------------------------------

N <- 1000
theta.hat <- NULL
for ( i in 1:1000){
x <- rcauchy(N)
theta.hat[i] <- mean(x > 1)}
Hm1.mean <- mean(theta.hat)
Hm1.var <- var(theta.hat)
```

```
# Hit or miss2 -----------------------------------------------------------

N <- 1000
theta.hat = NULL
for(i in 1:1000){
  x <- rcauchy(1000) %>% abs()
  theta.hat[i] <- 0.5*mean(x > 1)
}
Hm2.mean <- mean(theta.hat)
Hm2.var <- var(theta.hat)
```

```
# Hit or miss3 -----------------------------------------------------------

N <- 1000
theta.hat <- NULL
cauchy <- function(x){
  return(1/(pi*(1+x^2)))}
for(i in 1:1000){
  x <- runif(N)
  theta.hat[i] <- (1- mean(2*cauchy(x)))/2
  }
Hm3.mean <- mean(theta.hat)
Hm3.var <- var(theta.hat)
```

```r
# Antithetic Variate -----------------------------------------------------

N <- 1000
AD = NULL
for( i in 1:1000){
    x <- runif(N/2)
    y <- 1 - x
    temp1 <- cauchy(x)
    temp2 <- cauchy(y)
    theta.hat[i] <-0.5 - 0.5*(mean(temp1)+mean(temp2))}
  AV.mean <- mean(theta.hat)
  AV.var <- var(theta.hat)
```

```r
# 對偶變量
N = 500
theta.hat = NULL
for( i in 1:1000){
x <- runif(N)
y <- 1 - x
temp1 <- cauchy(x)
temp2 <- cauchy(y)
theta.hat[i] <- 0.5 - 0.5*(mean(temp1) + mean(temp2))}
AV.mean <- mean(theta.hat)
AV.var <- var(theta.hat)
```

```r
# Importance Sampling ----------------------------------------------------

N <- 1000
hes <- function(x){
  return(1/(pi*(1 + x^(-2))))
}

theta.hat <- NULL
for(i in 1:1000){
  U <- runif(N)
  x <- 1/U
  theta.hat[i] <- mean(hes(x))
}
Is.mean <- mean(theta.hat)
Is.var <- var(theta.hat)

ID <- (Mc.var - Is.var)/Mc.var
```

```r
# Control variate ------------------------------------------------------
# 令另一個function為1/(1+x)
N = 1000
f <- function(x){
  return(1/(1+x))
}
cauchy <- function(x){
  return(1/(pi*(1+x^2)))
}
theta.hat = NULL
for(i in 1:N){
u <- runif(1000)
B <- f(u)
A <- cauchy(u)
a <- -cov(A,B) / var(B) #est of c*

x <- runif(N)
T1 <- cauchy(x)
theta.hat[i]<-  mean(T1 + a * (f(x) -log(2, base = exp(1))))
}
Cv.mean <- mean(theta.hat)
Cv.var <- var(theta.hat)
```

```r
# Stratified Sampling --------------------------------------------------
# 切5層

cauchy <- function(x){
  return(1/(pi*(1+x^2)))
}
N <- 1000
SS = NULL
for(i in 1:N){
x1 <- runif(N/5, 0, 0.2)
x2 <- runif(N/5, 0.2, 0.4)
x3 <- runif(N/5, 0.4, 0.6)
x4 <- runif(N/5, 0.6, 0.8)
x5 <- runif(N/5, 0.8, 1)
S1 <- mean(cauchy(x1))
S2 <- mean(cauchy(x2))
S3 <- mean(cauchy(x3))
S4 <- mean(cauchy(x4))
S5 <- mean(cauchy(x5))
SS[i] <- mean(c(S1, S2, S3, S4, S5))
}
SS.mean <- mean(SS)
SS.var <- var(SS)

tmp1 <- c(Mc.mean, Hm1.mean, Hm2.mean, Hm3.mean, AV.mean, Is.mean, Cv.mean, SS.mean)
tmp2 <- c(Mc.var, Hm1.var, Hm2.var, Hm3.var, AV.var, Is.var, Cv.var, SS.var)
table <- rbind(tmp1, tmp2)
rownames(table) <- c("Mean","Variance")
colnames(table) <- c("Monte-Carlo", "Hit or miss1", "Hit or miss2", "Hit or miss3", "Antithetic",
                     " Importance", "Control", "Stratified(5)")
table
```

2.

```r
set.seed(15) # 3 + 12
# define function
Q2 <- function(x){
 y <- (exp(x)-1)/(exp(1)-1)
 return(y)}

x = seq(0,1,by = 0.001)
fx = Q2(x)
data <- data.frame(x = x, fx = Q2(x))
ggplot()+
  geom_line(data = data,aes(x = x, y = fx))+
  geom_vline(xintercept = 1,linetype = 2)+
  geom_ribbon(data = data, aes(x = x,ymin = 0, ymax = fx), fill = "#00FF00",alpha = 0.7)+
  labs(title = "Question 02")
```

```r
# Monte-Carlo Integration ---------------------------------------------------

theta.hat = NULL

for(i in 1:100){
x <- runif(10000)
theta.hat[i] <- mean(Q2(x))}
Mc.mean <- mean(theta.hat)
Mc.var <-var(theta.hat)
```

```r
N = 1000
theta.hat = NULL
for(i in 1:N){
x <- runif(N/2)
y <- 1 - x
temp1 <- mean(Q2(x))
temp2 <- mean(Q2(y))
theta.hat[i] <- 0.5*(temp1+temp2)}
AV.mean <- mean(theta.hat)
AV.var <- var(theta.hat)
```

```r
# Importance Sampling ---------------------------------------------------

theta.hat1 = NULL
theta.hat2 = NULL
theta.hat3 = NULL
theta.hat4 = NULL
for( i in 1:N){

  u <- runif(N)     #f3, inverse transform method
  x <- - log(1 - u * (1 - exp(-1)))
  fg <- Q2(x) / (exp(-x) / (1 - exp(-1)))
  theta.hat1[i] <- mean(fg)

  u <- runif(N/2)     #f3, inverse transform method + Antithetic Variate
  v <- 1 - u
  x1 <- - log(1 - u * (1 - exp(-1)))
  x2 <- - log(1 - v * (1 - exp(-1)))
  fg1 <- Q2(x1) / (exp(-x1) / (1 - exp(-1)))
  fg2 <- Q2(x2) / (exp(-x2) / (1 - exp(-1)))
  fg <- 0.5*(fg1+fg2)
  theta.hat2[i] <- mean(fg)

  u <- runif(N)    #f4, inverse transform method
  x <- tan(pi * u / 4)
  fg <- Q2(x) / (4 / ((1 + x^2) * pi))
  theta.hat3[i] <- mean(fg)
  u <- runif(N/2)    #f4, inverse transform method + Antithetic Variate
  v <- 1 - u
  x1 <- tan(pi * u / 4)
  x2 <- tan(pi * v / 4)
  fg1 <- Q2(x1) / (4 / ((1 + x1^2) * pi))
  fg2 <- Q2(x2) / (4 / ((1 + x2^2) * pi))
  fg <- 0.5*(fg1+fg2)
  theta.hat4[i] <- mean(fg)
}

mean11 <- c(mean(theta.hat1 ),mean(theta.hat2 ),mean(theta.hat3 ),mean(theta.hat4 ))
var11 <- c(var(theta.hat1 ),var(theta.hat2 ),var(theta.hat3 ),var(theta.hat4))
Is.mean1 <- mean(theta.hat3 )
Is.mean2 <- mean(theta.hat4)
Is.var1 <- var(theta.hat3)
Is.var2 <- var(theta.hat4)
```

```r
f <- function(x){
  x}
theta.hat = NULL
for( i in 1:N){
  u <- runif(10000)
  B <- f(u)
  A <- Q2(u)
  a <- -cov(A,B) / var(B)
  u <- runif(N)
  T1 <- Q2(u)
  T2 <- T1 + a * (f(u) - 1/2)
  theta.hat[i] <- mean(T2)
  }
Cv.mean <- mean(theta.hat)
Cv.var <- var(theta.hat)
```

```r
Q2 <- function(x){
  y <- (exp(x)-1)/(exp(1)-1)
  return(y)}
N <- 1000
SS = NULL
for(i in 1:N){
  x1 <- runif(N/5, 0, 0.2)
  x2 <- runif(N/5, 0.2, 0.4)
  x3 <- runif(N/5, 0.4, 0.6)
  x4 <- runif(N/5, 0.6, 0.8)
  x5 <- runif(N/5, 0.8, 1)
  S1 <- mean(Q2(x1))
  S2 <- mean(Q2(x2))
  S3 <- mean(Q2(x3))
  S4 <- mean(Q2(x4))
  S5 <- mean(Q2(x5))
  SS[i] <- mean(c(S1, S2, S3, S4, S5))
}
SS.mean <- mean(SS)
SS.var <- var(SS)
```

3.

```r
Q3<-function(x){
  (312.5/60)*exp((-1/5)*x)-
    (640/60)*exp((-1/4)*x)+
    (405/60)*exp((-1/3)*x)+
    (1/24)*exp((-1)*x)+
    (-4/3)*exp((-1/2)*x)
}
```

```r
set.seed(15) # 3 + 12
temp2 <- NULL
for(j in 1:1000){
  temp <- NULL
  for( i in 1:1000){
    tmp <- rexp(5)
    temp[i] <- sum(tmp*c(1:5))}
  temp2[j] <- sum(temp>21.6)/1000
}
Hm.mean <- mean(temp2)
Hm.var <- var(temp2)
```

```r
N <- 1000
theta.hat <- NULL
for(i in 1:1000){
  x <- runif(N,0,21.6)
  theta.hat[i] <-1 -  mean(Q3(x)*21.6)
}
Mc.mean <- mean(theta.hat)
Mc.var <- var(theta.hat)
```

```r
N <- 1000
theta.hat <- NULL
for(i in 1:1000){
  x <- runif(N/2,0,21.6)
  y <- 21.6 - x
  theta.hat[i] <- (2 - mean(Q3(x)*21.6) - mean(Q3(y)*21.6))*0.5
}
Av.mean <- mean(theta.hat)
Av.var <- var(theta.hat)
```

```r
N <- 1000
theta.hat <- NULL
for(i in 1:1000){
  tmp <- c()
  for(j in 1:5){
    x <- runif(N/5,(j - 1)/5*21.6,j/5*21.6)
    tmp <- c(tmp, Q3(x)*21.6)
  }
  theta.hat[i] <- 1 - mean(tmp)
}
SS.mean <- mean(theta.hat)
SS.var <- var(theta.hat)
```

4.

```r
library(dplyr)
library(ggplot2)
library(magrittr)
library(tidyverse)
```

```r
# First, simulate 100 observations from a mixed distribution of beta(2,3),
# each with probability 0.5. Then, use at least 3 density estimating methods
# to smooth the observations.
# You need to specify the parameters in the smoothing methods,
# and compare the results.

# 取出我們要的亂數
set.seed(106354012)
m = 100
x = rbeta(m,2,3)
# 畫個點的散佈圖
stripchart(x,pch=16,cex=0.5,col=3,main="Dotplot")
```

```r
# 然後畫個直方圖
y = seq(0,1,length=100)
hist( x, breaks = 10,probability = T,ylim = c(0,3),xlim= c(0,1),col = "#AAAAAA",main = "Density Estimation (h=
0.1)")
# 這是真實的beta(2,3)圖形
lines(y , dbeta(y,2,3) , col = "#00AAFF" , lwd = 3)
legend("topright" , "Beta(2,3)" , lty = 1 , col="#00AAFF" , lwd = 3)
# 這是直接用套件模擬出來的函數值
kernal_g = density(x, kernel = c("gaussian"), width = 0.1)
kernal_r = density(x, kernel = c("rectangular"), width = 0.1)
kernal_t = density(x, kernel = c("triangular"), width = 0.1)
# 把他們畫在plot裡面囉
lines(kernal_g, col = "#FF0000" , lwd = 2)
lines(kernal_r, col = "#00FF00" , lwd = 2)
lines(kernal_t, col = "#FFFF00" , lwd = 2)
legend("right",legend=c("gaussian","rectangular","triangular"), col=c("#FF0000","#00FF00","#FFFF00"),lwd=2,cex
=1)
box()
```

```r
# (h=0.2)
hist( x, breaks = 5,probability = T,ylim = c(0,3),xlim= c(0,1),col = "#AAAAAA",main = "Density Estimation (h=
0.2)")
# 這是真實的beta(2,3)圖形
lines(y , dbeta(y,2,3) , col = "#00AAFF" , lwd = 3)
legend("topright" , "Beta(2,3)" , lty = 1 , col="#00AAFF" , lwd = 3)
# 這是直接用套件模擬出來的函數值
kernal_g = density(x, kernel = c("gaussian"), width = 0.2)
kernal_r = density(x, kernel = c("rectangular"), width = 0.2)
kernal_t = density(x, kernel = c("triangular"), width = 0.2)
# 把他們畫在plot裡面囉
lines(kernal_g, col = "#FF0000" , lwd = 2)
lines(kernal_r, col = "#00FF00" , lwd = 2)
lines(kernal_t, col = "#FFFF00" , lwd = 2)
legend("right",legend=c("gaussian","rectangular","triangular"), col=c("#FF0000","#00FF00","#FFFF00"),lwd=2,cex
=1)
box()
```

```r
# 這是不想用套件的人寫的code~

set.seed(106354012)
x <- rbeta(100,2,3)
h=0.1

# histogram density estimator (直方圖)

histogram = function(x,h){
  g = function(x,a,b){
    if (x<=b & x>=a) {return(1)}
    else {return(0)}}          # 寫出一個計算個數前需要的函數
  n <- length(x)
  seq <- seq(min(x),max(x),by=h) # 從(0,1)間隔h做出切割點
  a = seq[-length(seq)]          # 每組下界
  b = seq[-1]                    # 每組上界
  ni = NULL
  for (k in 1:length(a)){
    ni[k] = sum(x<=b[k] & x>=a[k])}
  y_hat = NULL
  for (i in sort(x)){
    I=NULL
    for (j in 1:length(a)){
      gi=g(i,a[j],b[j])
      I=c(I,gi)}                 # 指標函數
    y=1/n*sum(ni/h*I)            # 模擬樣本之函數估計值
    y_hat=c(y_hat,y)}
  return(y_hat)}
```

```r
# naive density estimator
naive =function(x,h){
  w = function(y){
    if (abs(y)<1) {return(1/2)}
    else {return(0)}}             # 寫出公式裡的函數w
  n = length(x)
  seq = seq(min(x),max(x),length=length(x))
  y_hat = NULL
  for (i in seq){
    W=NULL
    for (j in x){
      wi=w((i-j)/h)
      W=c(W,wi)}
    y=1/n*sum(1/h*W)              # 樣本之函數估計值
    y_hat=c(y_hat,y)}
  return(y_hat)}

# kernel density estimator
# norm
kernel_norm=function(x,h){
  w=function(y){dnorm(y)}         # 核密度函數(常態)
  n = length(x)
  seq = seq(min(x),max(x),length=length(x))
  y_hat = NULL
  for (i in seq){
    W=NULL
    for (j in x){
      wi=w((i-j)/h)
      W=c(W,wi)}
    y=1/n*sum(1/h*W)
    y_hat=c(y_hat,y)}
  return(y_hat)}
```

```r
# Gamma核函數
kernel_gamma=function(x,h){
  w=function(y){dgamma(y,shape = 1)}           # 核密度函數(Gamma)
  n = length(x)
  seq = seq(min(x),max(x),length=length(x))
  y_hat = NULL
  for (i in seq){
    W=NULL
    for (j in x){
      wi=w((i-j)/h)
      W=c(W,wi)}
    y=1/n*sum(1/h*W)
    y_hat=c(y_hat,y)}
  return(y_hat)}

# 將亂數帶入函數找出估計函數值(h=0.1)
y1 <- histogram(x,0.1)              #h=0.1，也可使用其他h值
y2 <- naive(x,0.1)                  #h=0.1，也可使用其他h值
y3 <- kernel_norm(x,0.1)            #h=0.1，也可使用其他h值
y4 <- kernel_gamma(x,0.1)           #h=0.1，也可使用其他h值
```

```r
xx <- sort(x)
yy <- dbeta(xx,2,3)
y2 <- naive(xx,0.1)
y3 <- kernel_norm(xx,0.1)
y4 <- kernel_gamma(xx,0.1)
data <- cbind(xx,yy,y2,y3,y4) %>% as.data.frame()
colnames(data) <- c("sample","Beta(2,3)","naive","kernal(norm)","kernal(gamma)")
LBJ <- gather(data,key = "type",value = "value",2:5)
colnames(LBJ) <- c("sample","LineType","value")
library(magrittr)
LBJ$LineType %<>% as.factor()
library(ggplot2)
ggplot(data = LBJ) + labs(title = "Density Estimation (h=0.1)")+
  xlim(0,1)+ ylim(0,2.5) +
  geom_histogram(mapping = aes(x=sample,y=..density..),color="black",fill="gray",binwidth = 0.1)+
  geom_line(mapping = aes(x=sample,y=value,color=LineType,group=LineType),size=1.2)+
  theme(legend.title = element_text(colour="royalblue", size=20, face="bold"))+
  theme(legend.text = element_text(size = 16))+
  theme(legend.position = c(0.87,0.6))+
  theme(legend.background = element_rect(fill="#FFFFF0",size=1, linetype="solid",colour ="darkblue"))+
  theme(panel.grid.major = element_blank())+
  theme(panel.grid.minor = element_blank())+
  theme(panel.background = element_rect(fill="#EEFFFF",colour="black",size = 2))
```

5.

```
library(plyr)
library(dplyr)
library(tidyverse)
library(ggplot2)
library(KernSmooth)
library(broom)
library(magrittr)
library(igraph)
```

```
# kernel smooth
set.seed(106354012)

a <- seq(0,2*pi, length=100)
b <- sin(a) + rnorm(100,0,0.09)
c <- sin(a)

# kernel smooth (kernel is norm)
data <- ksmooth(a, b, kernel = "normal", bandwidth = 0.1)%>% as.data.frame()
data <- cbind(b,c,data)%>% as.data.frame()
ggplot(data,aes(x=x))+ labs(title="kernel smooth of sin(x) [h=0.1]",x="x",y="sin(x)")+
  geom_point(aes(y=b),col="#333344")+
  geom_vline(xintercept = 0,size=1)+
  geom_hline(yintercept = 0,size=1)+
  geom_line(aes(y=y),col="#00DDFF",lwd=1)+
  scale_x_continuous(breaks = c(0:2*pi))+
  theme(panel.grid.major = element_line(NA),panel.grid.minor =element_line(NA))+
  theme(panel.background = element_rect(color = "black",size = 2))+
  theme(plot.title = element_text(size = 30, face = "bold"))+
  theme(legend.title=element_text(size=24))+
  theme(legend.text=element_text(size=20))
```

```
# MSE
MSE <- c()
for (i in 1:1000) {
  b <- NULL
  b <- sin(seq(0,2*pi, length=100)) + rnorm(100,0,0.09)
  b1 <- ksmooth(a, b, kernel = "normal", bandwidth = 0.1)$y
  MSE[i] <- mean((b1-c)^2)
}
MSE <- mean(MSE) ; MSE
```

```r
# spline

# plot
df_values <- c(4, 9, 32, 64)

x = seq(from=0, to=2*pi, length=100)
f_x = sin(x)
epsilon = rnorm(100, 0, sd = 0.3)
y = f_x + epsilon

values <- data_frame(
  x = seq(from=0, to=2*pi, length=100),
  f_x = sin(x),
  epsilon = rnorm(100, 0, sd = 0.3),
  y = f_x + epsilon
)
overall <- NULL
for(df in df_values){
  overall <- smooth.spline(values$x, values$y, df=df) %>%
    augment() %>%
    mutate(df=df) %>%
    bind_rows(overall)
}
```

```r
overall <- cbind(overall,f_x) %>% as.data.frame()
multiple_df <- overall %>%
  ggplot(aes(x=x)) +
  geom_point(aes(y=y))+
  geom_line(aes(y=.fitted,color="#123456"),size=1) +
  facet_wrap(~df, nrow=2) +
  labs(title="Splines fit w / different degrees of freedom")+
  theme(legend.title=element_blank())+
  theme(legend.text = element_blank())+
  theme(panel.background = element_rect(colour = "black"))+
  geom_vline(xintercept = 0,size=1)+
  geom_hline(yintercept = 0,size=1)+
  theme(panel.grid.major = element_line(NA),panel.grid.minor =element_line(NA))+
  theme(panel.background = element_rect(color='#000000',size=2))+
  theme(plot.title = element_text(size = 30, face = "bold"))

multiple_df
```

```r
MSE <- c(1,1,1)
for (j in 4:64) {
  mse <- c()
  for (i in 1:1000) {
  b <- NULL
  b <- sin(seq(0,2*pi, length=100)) + rnorm(100,0,0.09)
  b1 <- smooth.spline(x = b,df=j)$y
  c <- sin(seq(0,2*pi, length=100))
  mse[i] <- mean((b1-c)^2)
  }
  MSE[j] <- mean(mse)
}
# MSE比較
MSE
```

```r
# Lowess
set.seed(106354012)

a <- seq(0,2*pi, length=100)
b <- sin(a) + rnorm(100,0,0.09)
c <- sin(a)

# 計算f值為0.23
lowess(x = a, y = b, f = 0.23)
```

```r
# plot
data <- lowess(x = a, y = b, f = 0.23) %>% as.data.frame()
data <- cbind(b,c,data)%>% as.data.frame()
ggplot(data,aes(x=x))+ labs(title="lowess smooth of sin(x) [h=0.1]",x="x",y="sin(x)")+
  geom_point(aes(y=b))+
  geom_vline(xintercept = 0,size=1)+
  geom_hline(yintercept = 0,size=1)+
  geom_line(aes(y=y),col="#00D0FF",lwd=1)+
  scale_x_continuous(breaks = c(0:2*pi))+
  theme(panel.background = element_rect(colour = "black",size=2))+
  theme(panel.grid.major = element_line(NA),panel.grid.minor =element_line(NA))+
  theme(plot.title = element_text(size = 30, face = "bold"))+
  theme(legend.title=element_text(size=24))+
  theme(legend.text=element_text(size=20))
```

```r
# MSE
MSE <- c()
for (i in 1:1000) {
  b <- NULL
  b <- sin(seq(0,2*pi, length=100)) + rnorm(100,0,0.09)
  b1 <- lowess(x = a, y = b, f = 0.23)$y
  MSE[i] <- mean((b1-c)^2)
}
MSE <- mean(MSE) ; MSE
```

```r
# running mean
set.seed(106354012)

a <- seq(0,2*pi, length=100)
b <- sin(seq(0,2*pi, length=100)) + rnorm(100,0,0.09)
c <- sin(seq(0,2*pi, length=100))

mse = NULL
for (k in 1:20){
r <- running_mean(b, binwidth=k)
x = NULL
for(i in 1:(100-k+1)){
x[i] <- mean(a[i:(i+k-1)])
}
mse[k] <- mean((sin(x)-r)^2)
num = which(mse==min(mse))
}
mse
```

```r
data <- cbind(a[1:length(b)],b,c[1:length(b)])%>%as.data.frame()
colnames(data) <- c("x","running mean","sin(x)")
library(magrittr)
data2 <- gather(data,key = "type",value = "value",2:3)
data2$type %<>% as.factor()
ggplot(data2)+ labs(title = "Running mean of sin(x)")+
  theme(panel.grid.major=element_blank(),panel.grid.minor=element_blank())+
  xlim(0,2*pi)+ ylim(-1,1) +
  geom_vline(xintercept = 0,size=1)+
  geom_hline(yintercept = 0,size=1)+
  geom_line(mapping = aes(x=x,y=value,color=type,group=type),lwd=1.87)+
  geom_point(mapping = aes(x=x,y=value),color="blue",size=1)+
  theme(legend.text = element_text(size = 16))+
  theme(legend.position = c(0.8,0.8))+
  theme(legend.background = element_rect(size=0.5, linetype="solid",fill ="#FFFFF0",colour ="black"))+
  theme(panel.background = element_rect(color='#000000',size=2))+
  theme(plot.title = element_text(size = 30, face = "bold"))+
  theme(legend.title=element_text(size=24))+
  theme(legend.text=element_text(size=20))
```

```r
# 1,000 simulation runs ( 設定k=2 )

a <- seq(0,2*pi, length=100)
b <- sin(seq(0,2*pi, length=100)) + rnorm(100,0,0.09)
c <- sin(seq(0,2*pi, length=100))

# MSE
c <- sin((a[-1]+a[-100])/2)
for (i in 1:1000) {
b <- NULL
b <- sin(seq(0,2*pi, length=100)) + rnorm(100,0,0.09)
b1 <- running_mean(b, binwidth=2)
MSE[i] <- mean((b1-c)^2)
}
MSE <- mean(MSE) ; MSE
```