R：簡介與應用*

高一誠**

2009/11

1. 前言

市面上有許多方便的計量或統計套裝軟體，例如 Stata 與 EViews，還有運算能力強大的 Gauss 與 MATLAB。爲甚麼這堂課要介紹 R？其實，管老師並沒有規定大家要用 R，他只有建議大家不要用 Excel 來寫計量作業。R 的最大好處是免費公開，隨時有來自全球各地的聰明人去維護與發展 R，這道理跟 Wikipedia 線上百科一樣。這特性也代表著，一些最新的計量方法或估計，較可能在 R 找到；反之，Stata 或 EViews 會落後一段時間才納入新方法。另外，網路上可以找到許多介紹 R 的文章與手冊，皆是免費公開的；我會將 R 相關手冊燒成光碟交給班代，同學們再自行傳閱參考。

我是上管中閔老師的計量經濟理論四（2007 年）時才接觸 R，並且之前沒有修過程式設計的相關課程。這表示，R 簡單易學，大家都能用來寫作業甚至作研究。同時，這也意味著，我撰寫的程式不一定有效率，歡迎大家來信討論。不過，因爲電腦運算速度快速進步，有時符合個人直覺反而更重要。當然，正確性永遠是第一目標。

這在份講義中，第 2 與第 3 節先簡單介紹 R 的安裝與基本運算。然後，透過計量（第 4 節）與財務（第 5 節）的例子，大家將瞭解 R 的實用性。這些例子是我讀博士班前期撰寫的。我的經驗是，不管你的興趣是理論或實證，將 R 學好，對將來修課及研究將有莫大的幫助。在介紹 R 的簡單應用後，我將撰寫程式時的一些經驗及建議事項放在第 6 節；當然，個人經驗有限，且仍在不斷地嘗試錯誤與學習。講義最後，提供一些題目讓大家練習。

2. 安裝

連上網頁The R Project for Statistical Computing: http://www.r-project.org/

→ 畫面左邊找到 Download

→ CRAN 點下去，然後就會出現一堆網址，往下找到台大的下載點

→ 選http://cran.csie.ntu.edu.tw/

→ 選擇你電腦的作業系統，一般是 Windows

---

* 本文爲計量經濟學（一）課程補充講義。其中，安裝（第 2 節）與基本指令（第 3 節）的內容，分別摘錄自謝鈺偉與莊惠菁的講義。他們兩位是我修管老師課時之助教，我非常感謝他們當時認真地介紹R。他們的完整講義可在管中閔老師的網頁中找到（進入http://www.fin.ntu.edu.tw/，點選教學課程中 2007 計量經濟理論四Introduction to R以及 2009 財務應用實證R Note）。

**台大國際企業研究所博士班；聯絡方式: alpha_kao@yahoo.com.tw。

→ 選 base

→ 點選最新版本，目前是R 2.9.2 for Windows

→ 下載完成後，點兩下讓他開始安裝

→ 安裝完畢後進入R的程式選程式套件 → 安裝程式套件 → 往下選到Taiwan（Taipei）將所有packages選項全部反白起來（滑鼠按住第一個然後往下捲，把所有選項都用藍色括起來）→ 按確定。接下來程式會下載R的所有數學函式庫，這需要一段時間才會下載完。當你可以打字進去命令視窗代表已經更新完，這時候就可以關掉程式。R會問你save workspace image？請選yes並不要更改存檔位址。

3. 基本指令

本節簡單彙整常用的基本指令。因爲時間限制，我們在課堂上不會將這些指令全部輸入示範。同學們再安裝 R 之後，應該花一點時間將這些指令輸入一遍，並呼叫結果加以檢視。其中，# 之後的文字是說明用，就算有數學式，R 也不會執行。這些指令不用刻意去背，常用的自然就會記住，不常用的需要時再查即可。若同學們需要一些複雜計算，但這裡沒有對應的指令，請上 R 網站搜尋或請教其他高手。另外，R 有許許多多別人寫好的 package（例如 GMM packages），我們在此無法一一介紹；若將來研究需要，這些 packages 也都有說明檔可參考。對於初學者而言，應該嘗試任何計算或估計都要自己寫，連 OLS 也不要用現成的 lm 指令（雖然本節也有介紹）。這樣作的好處是，你將十分熟悉各種計算或估計的細節，而當你有一些新想法時，就有能力對現有的程式加以修改或創新。

3.1 基本運算

```
n <- 15                      # <- is assignment operator, assign num 15 to variable n
n                            # shows the value of n
n <- (15)                    # works also
15 -> n                      # same
n1 <- (-3)
n1_abs <- abs(n1)            # absolute value
n1_abs
name <- "pp4"
k1 = n+2
K1 = n                       # case sensitivity, K1 differ. to k1
k2 = k1*3
k3 = k1/2
k4 = k3+ rnorm(1,0,1)        # gen. 1 normal random sample with mean = 0, sd = 1.
ls()                           # list all variables
ls.str()                     # list details on varables and their types
rm(list=ls(all=TRUE))        # delete all variables in memory
ls.str()                     # list details on varables and their types

x <- c(0,1,2,3)              # concatenate numbers{0,..,3} to vari. x: 4*1
x_trans = t(x)               # create a 1*4 matrix(transpose x)
x
x_trans
y <- c(2,3,5,1)
y_trans= t(y)
```

```
mix1 <- cbind(x,y)              # combine colume
mix1
mix2 <- rbind(x_trans, y_trans)             # combine row
mix2
rm(list=ls(all=TRUE))                       # delete all variables in memory

y <- 1:6                                    # creat sequence y=[1,2,3,4,5,6]'
z <- 3
r <- "pp4"
w <- c("pp4", "kai", "ccc", "www")          # combine char.s
complex <- 3i                               # complex numbers
logic     <- TRUE
mode(y);mode(z);mode(r);mode(w);mode(complex);mode(logic);
# mode: numeric, character, complex(3\textit{i}) and logical(TRUE, FALSE)
length(y);length(z);length(r);length(w);length(complex);length(logic);          # length of variavle
rm(list=ls(all=TRUE))                       # delete all variables in memory

k5 <- 3/0                       # infinite number
k5
k6 <- exp(-k5)                   # u can work on a Inf number
k6
not_a_number <- (k5-k5)         # not an number(Inf-Inf: undefined in math)
not_a_number
rm(list=ls(all=TRUE))           # delete all variables in memory
```

3.2 製造數列與隨機變數

```
x <- 1:(6-1)                                # want vector 1 to 5, by incre.=1
x
z1 <- seq(from=1, to=5, by= 0.5)            # want a vec. from 1 to 5, by incre.= 0.5
z1
z2 <- seq(length=9, from=1, to=5)           # wnat a    1*9 vector from 1 to 5.
z2
z3 <- c(1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5)  # concatenate numbers{1,..,5}
z3
z4 <- rep(3,5)                              # rep num 3, 5 times
z4
z5 <-    scan()                             # directly input by keyboard, use "Enter" to end input
rm(list=ls(all=TRUE))                       # delete all variables in memory

runif(n, min=0, max=1)                      # uniform
rnorm(n, mean=0, sd=1)                      # Gaussian (normal)
rexp(n, rate=1)                             # exponential
rgamma(n, shape, scale=1)                   # gamma
rpois(n, lambda)                            # Poisson, "arriveals of default events"
rweibull(n, shape, scale=1)                 # Weibull
rcauchy(n, location=0, scale=1)             # Cauchy, "fat tail phonomia"
rbeta(n, shape1, shape2)                    # beta, "recovery rate modelling"
rt(n, df)                                   # Student (t) "fat tail phonomia"
rchisq(n, df)                               # Chi-square
rbinom(n, size, prob)                       # binomial "tree pricing"
rmultinom(n, size, prob)                    # multinomial
rgeom(n, prob)                              # geometric
rhyper(nn, m, n, k)                         # hypergeometric
rlogis(n, location=0, scale=1)              # logistic    "transform the domain of variables"
rlnorm(n, meanlog=0, sdlog=1)    # lognormal "Geometric Brownian motion of equity price process"
rnbinom(n, size, prob)                      # negative binomial
```

```
rm(list=ls(all=TRUE))                    # delete all variables in memory
```

3.3 向量與矩陣

```
x <- 1:5
x
x[3]
x[3] <- 20                    # replace the third elt by 20
x

m3 = matrix(1:6, 2, 3)
m3
m3[,3]                        # the third colume
m3[2,]                        # the second row
m3[1,1]<- 100                 # replace (1,1) elt by 100
m3

x <- 1:10
x
x[x>5] <- 12                  # replace [6,7,8,9,10] by [12,12,12,12,12]
x
x[x%%2==0]                    # find out numeric no. that can be divided by 2 totally

y <- c("s", "pp", ".")        # useful when deal with missing values
y
y[y=="."] <- "missing_value"
y
rm(list=ls(all=TRUE))

x <- c(1,2,5,7,9)
sum(x)
prod(x)
max(x)
min(x)
which.max(x)                  # returns the index of the greatest element of x
which.min(x)                  # returns the index of the smallest element of x
range(x)
length(x)                     # number of elements in x
mean(x)
median(x)
var(x)                        # variance of the elements of x (calculated on n - 1)
round(x, 2)                   # rounds the elements of x to n decimals
rev(x)                        # reverses the elements of x
sort(x)                       # sorts the elements of x in increasing order; to sort in
decreasing order:
rev(sort(x))
rank(x)                       # ranks of the elements of x
rm(list=ls(all=TRUE))

m1 = matrix(data=5, nr=2, nc=2)      # gen a 2*2 matrix filled with num. 5
m2 = matrix(5, nr=2, nc=2)           # same work
m3 = matrix(1:6, 2, 3)               # gen a 2*3 matrix from top to butom, left to right, filled
num 1:6
m4 = matrix(1:6, 2, 3, byrow=TRUE)   # gen a 2*3 matrix from left to right, top to bottom,
filled num 1:6
m1
m2
```

```
m3
m4
apply(m3, 2, max)                    # apply "max" to m3's "2:colume-dimersion"
apply(m3,1, min)                     # apply "min" to m3's "1:row-dimersion"
rm(list=ls(all=TRUE))                # delete all variables in memory

m1 <- matrix(1, nr = 2, nc = 3)
m2 <- matrix(2, nr = 1, nc = 3)
rbind(m1, m2)                        # row-combine
m3 <- matrix(3, nr = 3, nc = 2)
m4 <- matrix(2, nr = 3, nc = 1)
cbind(m3, m4)                        # colume-combine
a = 10
x <- 1:4
z <- a*x                             # scalar times a vector
z
m5 = m3%*%m1                         # matrix multiple
m5
diag(m5)
rm(list=ls(all=TRUE))                # delete all variables in memory
```

3.4 重複運算

```
# basic control structures: for, if and while,
# basic logic operator: "&":AND; "|": OR; ">="; "=="
# for(i in 1:N ){...}
# if (conditions){...}
# while(conditions){...}

# example 1 : for
a = c(1,3,5,3,2)
b <- numeric(3)                      # initialized b as 3x1 vector of zeros
b
for(i in 1:length(b)){
      b[i] <- a[i+2]*2
}                                    # replace sth.
b
rm(list=ls(all=TRUE))                # delete all variables in memory

# example 2: if
a = c(1,3,5,3,2)
b <- c(1, 1000,7,9)
if(b[1]>3){
    a[1]<- 9
}                                    # if holds, excute {statement}
a
if(b[2]>999){
    a[1]<- 9
}                                    # if not holds, do nothing
a
rm(list=ls(all=TRUE))                # delete all variables in memory

# example 2-1: if
a = c(1,3,5,3,2)
b <- c(1, 1000,7,9)
if((b[1]>3)|(b[2]>999)){
    a[1]<- 9
```

```
 }                                                  # if    b[1]>3 OR b[2]>999, holds, excute {statement}
a
rm(list=ls(all=TRUE))

a = c(1,3,5,3,2)
b <- c(1, 1000,7,9)
if((b[1]>3)&(b[2]>999)){
    a[1]<- 9
 }                                                  # if    b[1]>3 AND b[2]>999, holds, excute
{statement}
a
rm(list=ls(all=TRUE))                              # delete all variables in memory

# example 3: for and if else

a <- c(1,2,2,4)
b <- c(1,3,2,1)
c <- numeric(4)

for(i in 1:length(a)){
   if (a[i]== b[i])
         c[i] <- a[i]
   else
         c[i] <- 0
 }
 rm(list=ls(all=TRUE))

# example 3-1: for and if else (equivalent to example 3)
a <- c(1,2,2,4)
b <- c(1,3,2,1)
c <- numeric(4)
c[a==b] <- a
c[a!= b] <- 0
c
rm(list=ls(all=TRUE))

# example 4 : change dta's row order by a specific pattern
original_data = matrix(1:30, nr=10, nc=3)
original_data
dumy = dim( original_data )
row_dim <- dumy[1]
colume_dim <- dumy[2]

new_row_index= round(runif(row_dim,1,row_dim))       # want random row inex,iid, draw with
replacement)
new_row_index

new_data = matrix(0, row_dim, colume_dim)
for (i in 1:row_dim){
 new_data[i,] <- original_data[new_row_index[i],]    # replace ori-dta's row by new index, save to
new_dta
}

new_data
rm(list=ls(all=TRUE))
```

3.5 OLS

```
y<- rnorm(10)
```

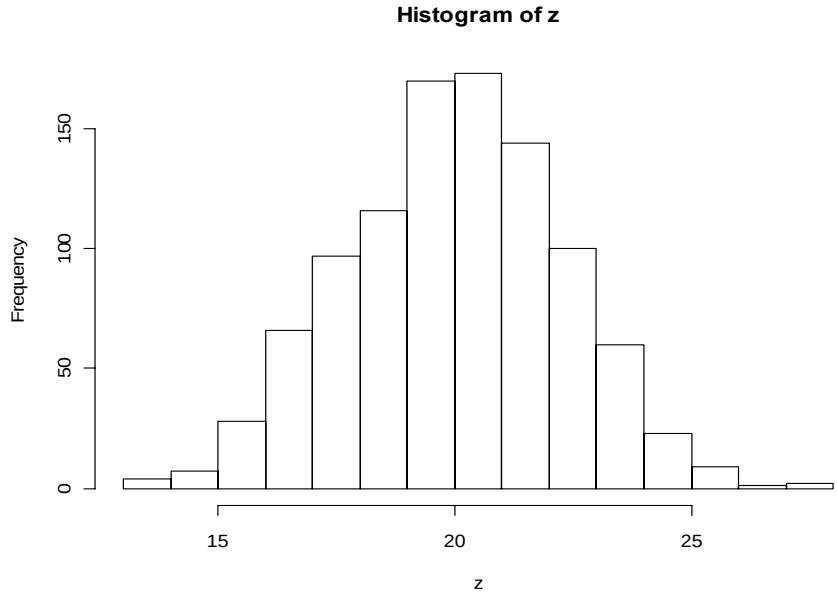
```
x<- rnorm(10)
z<- rnorm(10)
result_1 <-lm(y~x)              # with intercept
summary(result_1)
result_2 <-lm(y~x-1)            # without intercept
summary(result_2)
result_3 <-lm(y~x+z-1)          # without intercept
summary(result_3)
```

4. 計量上的簡單應用

由於講義目的是輔助同學有效地學習計量經濟學，所以課堂上的講解將著重於此部分。本節例子中，有些計算是可以使用 package 執行，但我們仍全部仔細寫出，這樣大家才能清楚瞭解背後的計算過程。

4.1 Normal distribution

We first generate a sequence with $T = 1000$ from the standard normal distribution. Then, another sequence of the length is generated from the normal distribution with mean of 10. It is clearly that the linear combination of normal distributions is still a normal one.



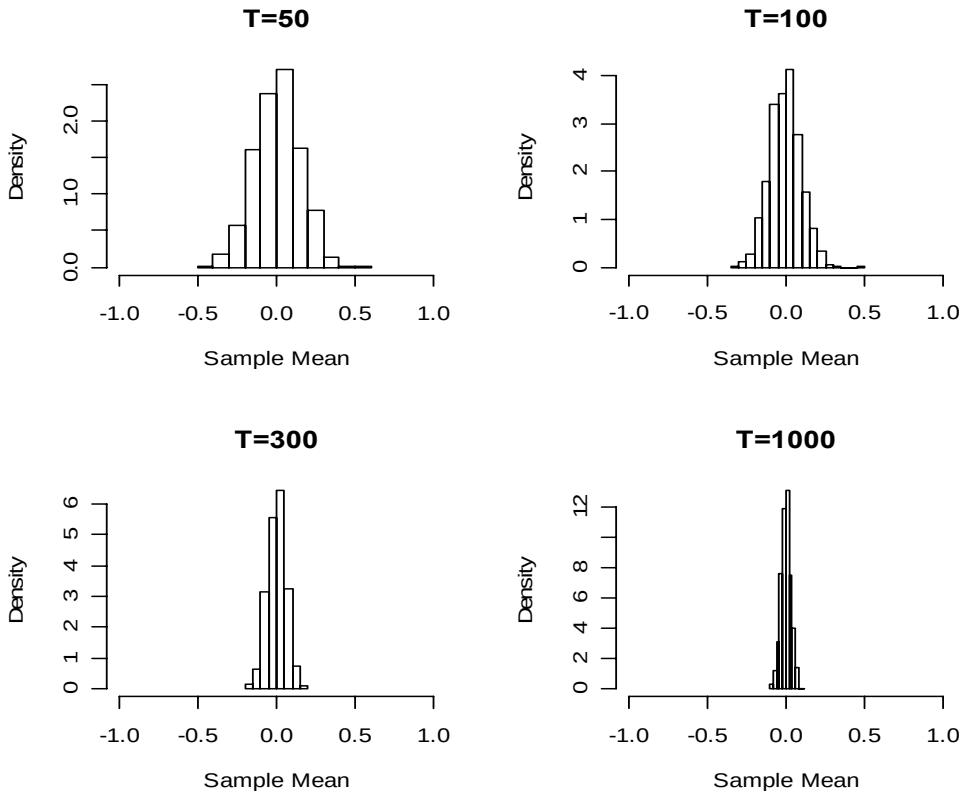
R code

```
x<- rnorm(1000)
plot(x)
hist(x)
mean(x)

y<- rnorm(1000, mean=10, sd=1)
z<- x+2*y
hist(z)
mean(z)
```

4.2 Law of large numbers

Random samples with numbers of $T = 50$, $T = 100$, $T = 300$, and $T = 1000$ are drawn form the standard normal distribution for 1000 times, and then their sample averages are computed each time. We plot the histograms of 1000 realizations for those cases as follows. For comparison, histograms are represented by the percentage scale of density. As the sample size increases, the sample averages approach the true mean of zero. This expresses the law of large numbers.



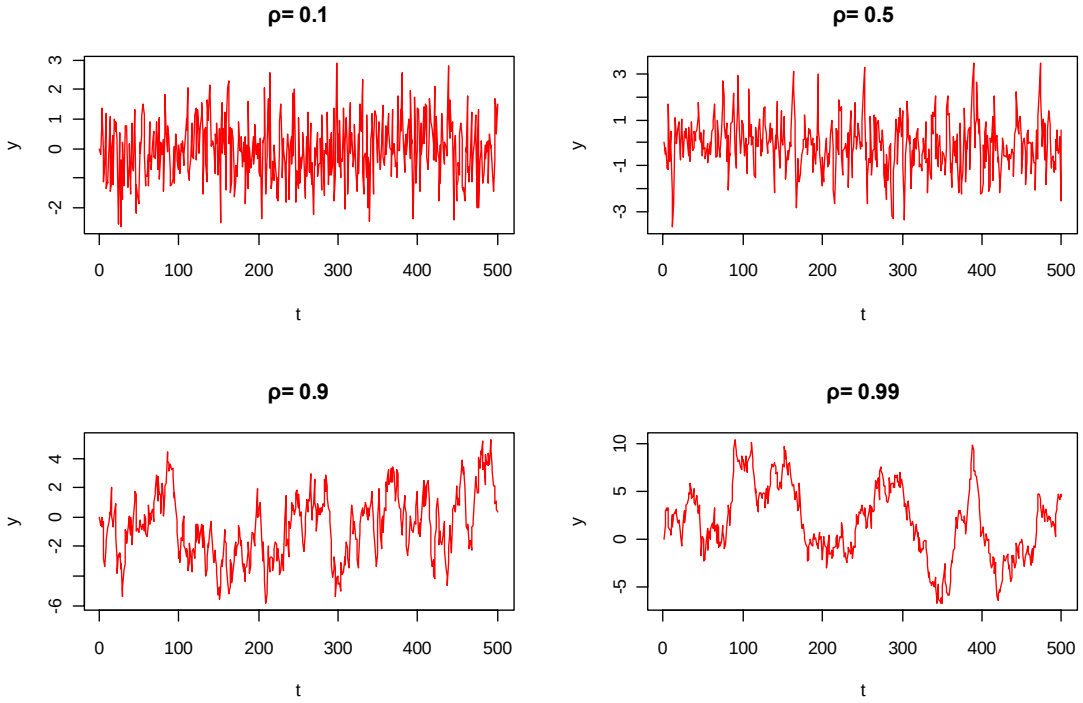
R code

```
fun<-function(t,n){
   x<-numeric(n)
   for(i in 1:n){
      x[i]<-mean(rnorm(t))}
   x}
# t stands for sample size
# n stands for repetitions

par(mfrow=c(2,2))
hist(fun(50,1000),xlim=range(-1,1),freq=FALSE,main='T=50',xlab='Sample Mean')
hist(fun(100,1000),xlim=range(-1,1),freq=FALSE,main='T=100',xlab='Sample Mean')
hist(fun(300,1000),xlim=range(-1,1),freq=FALSE,main='T=300',xlab='Sample Mean')
hist(fun(1000,1000),xlim=range(-1,1),freq=FALSE,main='T=1000',xlab='Sample Mean')
```

4.3 AR(1) process

The model of first-order autoregressive, AR(1), is specified as $y_t = \rho y_{t-1} + \varepsilon_t$. In this simulation, we simply let $y_1 = 0$ and then generate different cases of $\rho = 0.1, 0.5, 0.7, 0.99$ under $\varepsilon_t \sim N(0,1)$ for $t = 2,3,\cdots,500$.



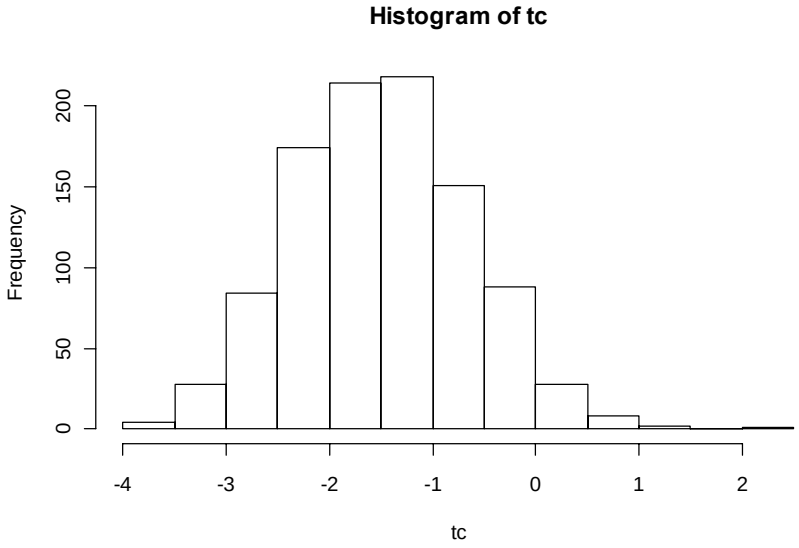
R code

```
n<- 500
ar<-function(r,n){
   y<- array(0,n)
      for (i in 2:n){
      y[i]<-r*y[i-1]+rnorm(1)}
   y
}

par(mfrow=c(2,2))
plot(ar(r=0.1,n), type="l",col="red",main=' ρ = 0.1',xlab='t',ylab='y')
plot(ar(r=0.5,n), type="l",col="red",main=' ρ = 0.5',xlab='t',ylab='y')
plot(ar(r=0.9,n), type="l",col="red",main=' ρ = 0.9',xlab='t',ylab='y')
plot(ar(r=0.99,n), type="l",col="red",main=' ρ = 0.99',xlab='t',ylab='y')
```

4.4 Simulation of t-values

We generate 1000 sequences of y_t such that $y_t = y_{t-1} + \varepsilon_t$ with $\varepsilon_t \sim N(0,1)$ for $t = 1,2,3,\cdots,500$. By running the OLS of $y_t = \alpha + \beta y_{t-1} + e_t$, we then obtain a sampling distribution of t-values under the null hypothesis of $\beta = 1$.



R code

```
# t stands for sample size
# n stands for repetitions
# beta is the true parameter
# ho is the null hypothesis
# Specification with constant term

fun<- function(t,n,beta,ho){
   tv<- numeric(n)
   for(i in 1:n){
      ty<- numeric(t)
      u<- rnorm(t)
      for(j in 1:t){
         ty[j]<- sum(u[1:j])}
      y<- ty[-1]
      x<- ty[-t]
      one<- matrix(1,nr=t-1,nc=1)
      xx<- cbind(one,x)
      xi<- solve(t(xx)%*%xx)
      b<- xi%*%t(xx)%*%y
      e<- y-xx%*%b
      v<- (t(e)%*%e)/(t-2)
      sd<- sqrt(v*xi[2,2])
      tv[i]<- (b[2,1]-ho)/sd}
   sort(tv)}
tc<- fun(t=500, n=1000,beta=1,ho=1)
hist(tc)
```

4.5 Q test of Ljung and Box (1978)

In this subsection, the Q test of Ljung and Box (1978) is applied to examine whether the time series is serially uncorrelated or not. For comparison, we adopt two different kinds of series, labeled as Y1 and Y2. Y1 is a random sequence from the standard normal distribution and Y2 is the term spread of US 10yr and 2yr Treasury notes, from Jun. 1976 to Jan. 2009. Inherently, Y1 is serially uncorrelated and hence it can serve as a detector for the correctness of the R

code presented later.

The term spread is adopted here because it contains useful information about future states of the economy. For example, Estrella and Mishkin (1997, European Economic Review 41, 1375-401) find that, in both the United States and Europe, the term spreads have significant predictive power for the real economic activity and inflation. In fact, in order to hedge the macroeconomic risk, investment banks have sold financial products with returns depending on the term spreads. Therefore, it should be interesting to examine the property of Y2 by a diagnostic test. The Ljung-Box Q statistic is computed as

$$T(T+2)\sum_{i=1}^{m}[\hat{\rho}(i)^2/(T-i)],$$

where m is a given number of autocorrelations. The statistic converges to $\chi^2(m)$ in distribution. In this example, we compute the statistic given $m \in [1, 50]$ for each series.

4.5.1 The result

In figure 1, Y1 is obviously a white noise and we can not reject the null hypothesis that the series is serially uncorrelated, which means that the code should be correct. Notice that as m increases, both the Q statistic and the critical value also increase. When $m = 50$, the critical value of significance at 95% level is 67.5. On the other hand, figure 1 also shows that Y2 is serially correlated since its Q statistic is pretty greater than the critical values in any case of m. This result favors investment bankers that if they could model the term spread properly, the corresponding risk of the financial product can be hedged as well.

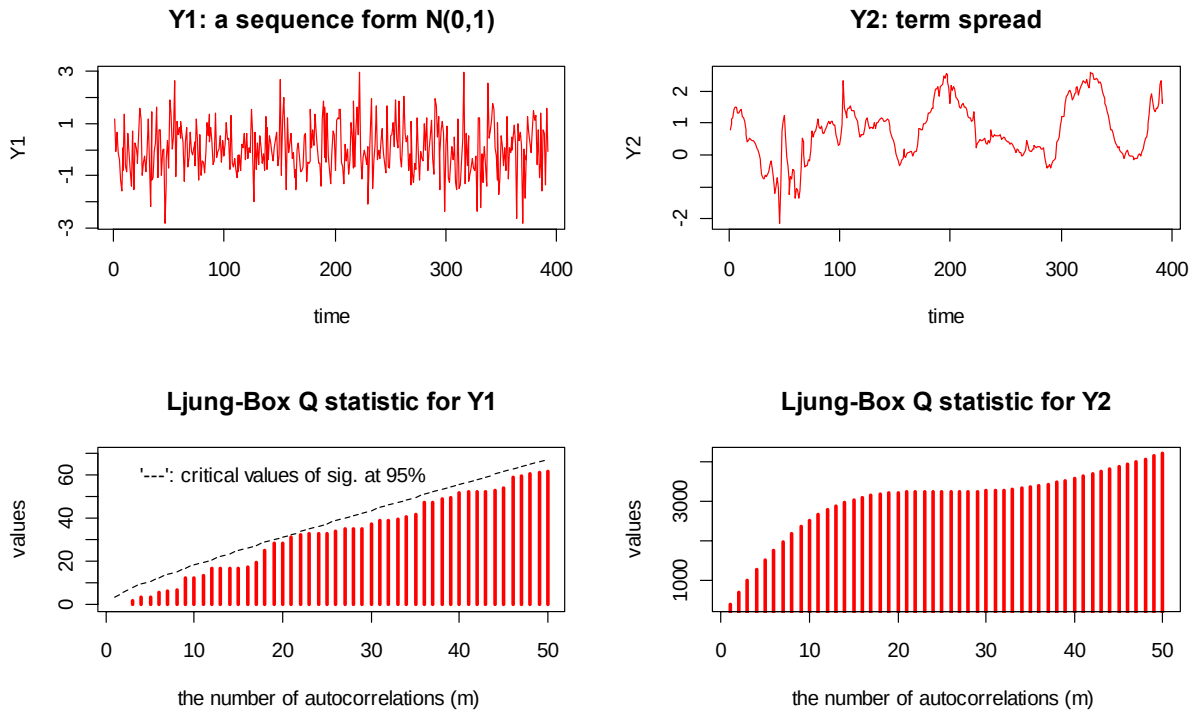


Figure 1. Q test of Ljung and Box (1978)

4.5.2 R code

```
data<- read.table("D:/Data/FEMA/hw1_US_rates.txt",header = TRUE)

# This code computes the modified Q test of Liung and Box (1978)
# "y2" is specified as the objective vector from "data" matrix
# "n" is the given number of autocorrelations, i.e., Ho: ρ(1)=...=ρ(n)=0

y2<- data[,4]
T<-length(y2)
y1<- rnorm(T)

Q_LB<- function(y,n){
Qc<- matrix(0,n,4)

for(m in 1:n){
ar<- function(x,i){
   x1<- x[1:(length(x)-i)]
   x2<- x[(1+i):length(x)]
   cor(x1,x2)
   }
am<- matrix(0,m,1)
for(i in 1:m){
   am[i,1]<- ((ar(y,i))^2)/(T-i)
   }
Q<- T*(T+2)*sum(am)
Qc[m,1]<- Q
Qc[m,2]<- qchisq(0.90, df=m)
Qc[m,3]<- qchisq(0.95, df=m)
Qc[m,4]<- qchisq(0.99, df=m)
   }
Qc}

Qc1<- Q_LB(y1,50)
Qc2<- Q_LB(y2,50)

par(mfrow=c(2,2), cex=1.1)

plot(y1,type="l",col="red",
main="Y1: a sequence form N(0,1)",
xlab="time", ylab="Y1")

plot(y2,type="l",col="red",
main="Y2: term spread",
xlab="time", ylab="Y2")

plot(Qc1[,1],type="h",col="red",lwd=3, ylim=c(0,70),
main="Ljung-Box Q statistic for Y1",
xlab="the number of autocorrelations (m) ", ylab="values");
text(20,60,"'---': critical values of sig. at 95% ");
lines(Qc1[,3],lty=2)

plot(Qc2[,1],type="h",col="red",lwd=3,
main="Ljung-Box Q statistic for Y2",
xlab="the number of autocorrelations (m) ", ylab="values");
lines(Qc2[,3],lty=2)
```

5. 財務上的簡單應用

上李存修老師的期貨與選擇權時，老師要我們評價一檔股權連結型商品。在風險與報價合理性部分，我是用 R 來分析。事實上，在風險量化後，可以進一步利用期貨市場來避險，但這部分與 R 程式較無關，就略過不提。

5.1 產品簡介

本例爲證券公司實際發行之股權連結型商品，契約條件如下：

連結標的	：	台塑普通股(1301.TW)
交易日/發行日	：	2003 年 8 月 15 日(預計)
評價日	：	2004 年 2 月 16 日(預計)
到期日	：	2004 年 2 月 18 日(預計，若遇非交易日，以下一交易日為到期日)
承作期間	：	六個月
契約金額(F)	：	最低新台幣壹拾萬元整(以新台幣伍萬元為一遞增單位)
發行價格	：	100%
期初參考價(S_0)	：	交易日台塑 (1301.TW)之收盤股價
期末參考價(S_T)	：	評價日台塑 (1301.TW)之收盤股價
上限價(H)	：	期初參考價之 115% (S_0 × 115%)
履約價格(K)	：	期初參考價之 100% (S_0 × 100%)
獲利加速乘數	：	168%
交割金額	：	契約金額(F) × 發行價格
手續費	：	0.5%
評價日結算方式	：	1. 到期時，若評價日連結標的收盤價大於或等於上限價格，則發行者支付 **F*{100%+1.68*[(H-K)/K]}** 2. 到期時，若評價日連結標的收盤價大於或等於履約價格且小於上限價格，發行者支付 **F*{100%+1.68*[(S_T -K)/K]}** 3. 到期時，若評價日連結標的收盤價小於履約價格，發行者支付 **(F/K)* S_T** 4. 以上結算金額，以元為最小單位並以小數點第一位四捨五入計算
提前贖回條款	：	提前贖回由台証另行報價，提出申請時間為每週三 12:30 前
最大可能損失	：	全部之交易價金
股價調整	：	連結標的辦理配發股息、紅利、增資時，履約價格之調整及其他相關事項之約定： S：除權(息)前一日收盤價　S'：權(息)參考價 K：調整前之履約價格　K'：調整後之履約價格 N：調整前之行使比例　N'：調整後之行使比例 R：現金增資每股認購價　m：現金增資認股率 n：無償配股率　T：發行人參與除權之稅賦 D：現金股利
		(一)盈餘轉增資、資本公積轉增資及現金增資： 1.除權參考價計算公式如下： $S' = \frac{S + mR + T}{1 + m + n}$ 2.調整後之行使比例 N' = N × (S / S') 3.調整後之履約價格 K' = K × (S' / S) (二)發放現金股利： 1.行使比例不調整 2.調整後之履約價格 K' = K× [S － D] / S

5.2 投資者的報酬與風險

本產品最低購買金額爲 10 萬元，換言之，至少購買 1904.76 單位（100000/52.5=1904.76）。然而，爲方便分析與比較，以下簡單假設某投資人用 52.5 元購買一單位的商品契約，故 $F = 52.5$。所有的分析結果，乘以適當倍數，而小數點再四捨五入後，就可以適用於不同額度的投資情況。對於投資者而言，其最終報酬完全決定於期末股價 S_t 落於哪一區間。我們分別計算如下。

（1）$S_t \geq 60.375$：已達連結標的上限價，所以投資者期末報酬均爲 65.73 元，因爲 $52.5 \times [1 + 1.68 * (60.375 - 52.5) / 52.5] = 65.73$。不管最後股價多高，這是此商品最大的可能報酬。

（2）$52.5 \leq S_t < 60.375$：此時投資者期末報酬爲 S_t 之遞增函數。當 $S_t = 52.5$，爲此區間股價的下界，期末報酬等於 52.5，因爲 $52.5 \times [1 + 1.68 * (52.5 - 52.5) / 52.5] = 52.5$。

（3）$S_t < 52.5$：投資者期末報酬就是 S_t，因爲 $(F / K) * S_t = (52.5 / 52.5) * S_t = S_t$。

因爲理論上股價可以趨近於 0，所以當 S_t 落於區間（3）並趨近於 0，投資者可能會損失所有的投資金額。

5.3 產品報價之合理性

在此，我們假設台塑股價走勢服從幾何布朗運動（geometric Brownian motion，以下簡稱 GBM），進一步來評價此商品的現值爲何。GBM 可表示如下：

$$\delta S = \mu S \delta t + \sigma S \varepsilon \sqrt{\delta t} \ ,$$

其中，S 是股價，δS 是股價的變動，μ 爲股價年化後的預期報酬，δt 爲股價變動之期間，σ 爲股價報酬率標準差（以下簡稱 Volatility），而 ε 爲服從標準常態分配之隨機變數。在風險中立的環境下，若期間無股利發放，μ 就爲無風險利率（risk-free interest rate，以下用符號 r 來代表此利率）。在美國市場，r 通常是用 3 個月的國庫券利率；然而，台灣的國庫券利率不具代表性，我們改以流動性較佳的 3 個月商業本票利率來替代。在 92 年 8 月 15 日，$r = 0.0133$。另外，依據 John Hull 的建議，估計 Volatility 可用與商品期間相同的最新歷史資料，因此，我們用 92 年 2 月 17 日至 92 年 8 月 15 日的台塑股價資料，來估計報酬率的日標準差。此標準差估計值經年化後（乘以 $\sqrt{252}$）爲 0.3377845。值得注意的是，這只是 Volatility 的歷史資料估計值，而估計的標準誤差爲 0.02127848（$= 0.3377845 / \sqrt{2 * 126}$）。在分析證券商的風險控管中，我們將會把 Volatility 的不確定性考慮進去。

至此，模擬 GBM 所需之參數都已給定。我們用統計軟體 R 來模擬 1 萬次的股價走勢與商品期末報酬，並將模擬結果彙整於圖 2。在圖 2A 中，可看出 S_t 十分接近一對數常態分配。我們先用 $52.5 \times [1 + 1.68 * (S_t - 52.5) / 52.5]$ 的公式將全部 S_t 加以轉換後（此

原始轉換尚未加入上限價與執行價的不同區間計算），得到圖 2B 的右厚尾分佈。換言之，此原始轉換若不加以設限，商品價值大幅增加的可能性很高。因此，發行者就用上限價 60.375 將上檔風險控制住；然後，以增加$52.5 \le S_t < 60.375$之區間報酬來吸引投資者。

對於投資者而言，此商品是否合理呢？我們依據契約規則，將本商品的期末價值模擬分配繪於圖 2C。可以清楚看到，本商品期末價值有許多機會達到上限價，因此，投資者本質上是放棄了一些潛在報酬，來換取$52.5 \le S_t < 60.375$區間之額外報酬。我們將 1 萬次的商品期末價值全部折現回定價日（就是發行日 92 年 8 月 15 日），並計算其算數平均值，得到 51.949125 元。這顯示出，在此模擬分析中，投資者是用 52.5 元來購買現值只有 51.949125 元的商品。這相當於用 1 元買到價值只有 0.9895 的商品。

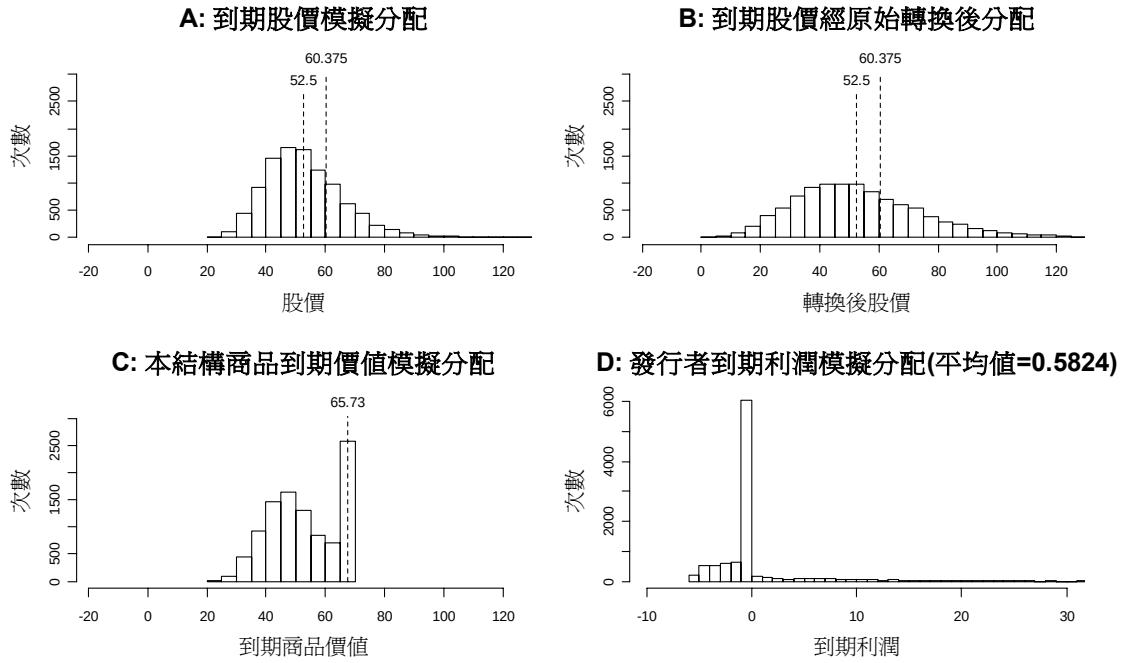


圖 2 股價與相關報酬模擬

5.4 證券商風險來源

若發行者一開始就將交易得來的 52.5 元去市場上換取一股，不管股價如何變化，期末時發行者均能賣股得到S_t元（假設台塑股票交易無流動性問題）。我們將所有模擬的S_t減去期末須付給投資人的金額（就是圖 2C 的商品到期價值），得到圖 2D 的發行者期末利潤分配圖，其平均值爲 0.5824。故在本模擬分析中，平均而言，發行此商品的券商是賺錢的。然而，由圖 2D 隱約可看出，平均值爲正是由於那些遠高於上限價之S_t的貢獻。然而，當 Volatility 值變低時，S_t高於上限價的機率也會降低，這就對發行者帶來風險。接下來，我們將焦點轉至 Volatility 的不確定性會帶給發行者那些損失。

在圖 3 中，我們以 Volatility 的估計值 0.3377845 爲中心，左右各延展 4 個估計標準誤差，並模擬此區間中不同 Volatility 下，發行者的期末利潤平均值。可清楚看出，當 Volatility 較小時，期末利潤可能爲負，而整體而言，期末利潤爲 Volatility 的遞增函數。事實上，我們可以對此結果進一步地拆解。因爲發行者一開始就將交易得來的 52.5 元去市場上換取一股，當期末股價 $S_t \le 52.5$ 時，發行者可賣得 S_t 元，而需付給投資者的金額就是 S_t ，故發行者不會有任何損失；另一方面，當 $S_t > 65.73$ 時，發行者需付給投資者的金額固定爲 65.73，故發行者可以獲得利潤。因此，當 $52.5 < S_T < 65.73$ 時，發行者將處於賠錢的狀態。從這個角度來看，圖 3 中 Volatility 較小時的期末預期損失來源，就是當股價的波動性不高，導致 S_t 有很大的可能性會落在 $52.5 < S_T < 65.73$ 之間，這就讓發行者賠錢的機率升高。反之，若 Volatility 很高時，期末股價處於 $S_t \le 52.5$ 或 $S_t > 65.73$ 的可能性大增，當 $S_t \le 52.5$ 時發行者不會有損失，而 $S_t > 65.73$ 時發行者則賺取正的利潤。

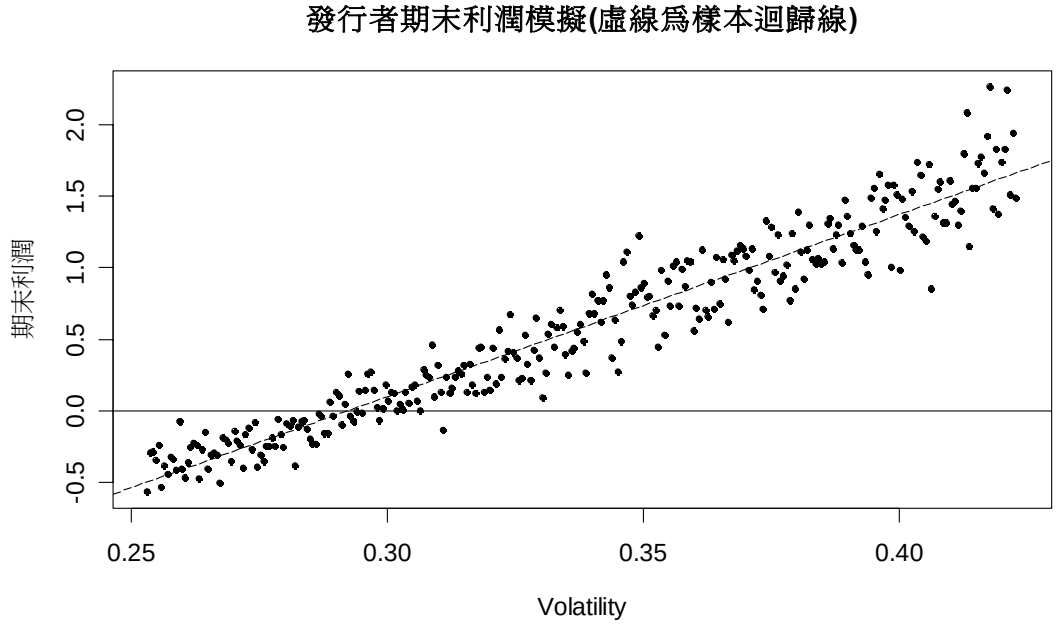


圖 3 不同 Volatility 下發行者的期末利潤

5.5 R 程式碼

5.5.1 評價此結構商品現值與相關圖形

```
# The price behavior here is a geometric Brownian motion.
# The input variables are as follows:
# So = initial stock price,
# m    = expected return
# r    = risk free interest rate
# v    = standard deviation of the return (i.e. the volatility),
```

```
# t    = the whole time interval (in terms of years)
# n    = the number of simulating samples

So<- 52.5
m<-   0.0133
r<-   0.0133
v<-   0.3377854
t<-   0.5
n<-   10000

# The final stock price St is generated from the following loop.
St<- matrix(0,nrow=n,ncol=1)
for(i in 1:n){
    e<- rnorm(1)
    GBM<- So*exp((m-(v^2)/2)*t+v*e*(t^0.5))
    St[i,1]<- GBM
    }

# We now evaluate the present value (pv) of the subject.
cSt<- ifelse(St<=60.375, St, 60.375)
cSt<- ifelse(cSt<=So, cSt, So*(1+1.68*((cSt-So)/So)))
pcSt<- cSt*exp(-r*t)
pv<- mean(pcSt)

# the risk control and profit distribution for the issuer
rSt<- So*(1+1.68*((St-So)/So))
pSt<- St-cSt
ppSt<- pSt*exp(-r*t)
pp<- mean(ppSt)

# plotting the results
par(mfrow=c(2,2), cex.main=1.8, cex.lab=1.5)
hist(St,br=n/400,xlim=range(So*(1-4*v),So*(1+4*v)),ylim=range(0,n/3),main='A: 到期股價模擬分
配',xlab='股價',ylab='次數');segments(So,0,So,n/3.76,lty=2);text(x=So,y=n/3.4,
labels="52.5");segments(60.375,0,60.375,n/3.3,lty=2);text(x=60.375,y=n/3, labels="60.375")
hist(rSt,br=n/300,xlim=range(So*(1-4*v),So*(1+4*v)),ylim=range(0,n/3),main='B: 到期股價經原始
轉換後分配',xlab='轉換後股價',ylab='次數');segments(So,0,So,n/3.76,lty=2);text(x=So,y=n/3.4,
labels="52.5");segments(60.375,0,60.375,n/3.3,lty=2);text(x=60.375,y=n/3, labels="60.375")
hist(cSt,br=n/800,xlim=range(So*(1-4*v),So*(1+4*v)),ylim=range(0,n/3),main='C: 本結構商品到期
價值模擬分配',xlab='到期商品價值',ylab='次數
');segments(65.73+2,0,65.73+2,n/3.3,lty=2);text(x=65.73+2,y=n/3, labels="65.73")
hist(pSt,br=n/200,xlim=range(-10,30),main='D: 發行者到期利潤模擬分配(平均值=0.5824)',xlab='到
期利潤',ylab='次數')
```

5.5.2 模擬不同 Volatility 下券商期末預期利潤

```
# We allow different values of volatility in this case

# The price behavior here is a geometric Brownian motion.
# The input variables are as follows:
```

```
# So = initial stock price,
# m    = expected return
# r    = risk free interest rate
# v    = standard deviation of the return (i.e. the volatility),
# t    = the whole time interval (in terms of years)
# n    = the number of simulating samples
# nv = the number of different volatilities

nv<- 300
profit<- matrix(0,nrow=nv,ncol=1)
Volatility<- matrix(0,nrow=nv,ncol=1)

for(j in 1:nv){

So<- 52.5
m<-   0.0133
r<-   0.0133
v<-   0.3377845-4*0.02127848+j*(8*0.02127848/nv)
t<-   0.5
n<-   1000

# The final stock price St is generated from the following loop.
St<- matrix(0,nrow=n,ncol=1)
for(i in 1:n){
    e<- rnorm(1)
    GBM<- So*exp((m-(v^2)/2)*t+v*e*(t^0.5))
    St[i,1]<- GBM
    }

# the final value of the subject.
cSt<- ifelse(St<=60.375, St, 60.375)
cSt<- ifelse(cSt<=So, cSt, So*(1+1.68*((cSt-So)/So)))

# the profit distribution for the issuer.
rSt<- So*(1+1.68*((St-So)/So))
pSt<- St-cSt
mp<- mean(pSt)

Volatility[j,1]<- v
profit[j,1]<- mp

}

# plotting the results
z <- lm(profit ~ Volatility)
par(mfrow=c(1,1), cex=1.4)
plot(Volatility,profit,pch=20,main='發行者期末利潤模擬(虛線爲樣本迴歸線)',xlab='Volatility',ylab='
期末利潤');abline(z,lty=5);abline(h=0,lty=1)
```

6. 一些建議

正確性，永遠是撰寫程式的第一目標，因此，我們要想辦法去確認程式的正確性。當程式邏輯不通或格式錯誤時，執行 R 會顯現錯誤訊號；不過，有時錯誤的程式卻可以完整執行，這時，驗證就很重要了。一個驗證的簡單方法是，找出理論上的特例，而此特例有明確可預測的結果。若程式執行特例時符合此預測結果，那麼出錯的可能性就少很

多。例如，在 4.4 中，我們造一個隨機序列 Y1，就是要驗證程式的正確性。當然，若有很多特例可以驗證，那麼就該多試試。另外，對於不熟悉的程式指令，請務必造一些簡單例子，來確認其結果與你預期相同。

當在撰寫大型程式時，盡可能將程式分成許多模組。如此一來，就可以針對個別模組來檢查程式的正確性。例如，4.3 的 function 其實就是 OLS；此時，我可以造兩個隨機序列，然後先用 R 內建的 lm 執行 OLS，再比較其結果是否與 function 一致。確認 function 的正確性後，下次需要時就可以依此基礎加以改造變化。當然，改造變化後還是要再確認正確性。模組化還有另一個好處，就是可以讓我們有效率地分配時間。當你撰寫大型程式時，由於十分專注，時間會過的很快，有時一個早上過了，眼睛都還沒休息。如果將程式的藍圖事先想好（或是在紙上先畫好計算程序的分鏡腳本），並適當地分割成不同模組，完成一個模組後，我們就能先休息一下。講義中的例子皆是短程式。我最近撰寫了一組含 5865 字元的程式，此時，就一定要適當地分割模組。

由於程式架構是建立在個人抽象思考上，而每個人的思考風格皆不盡相同，所以要看懂別人的程式很耗時間。其實，就算是自己寫的程式，一段時間後就會忘記其中細節，因此，在程式中加入文字描述是絕對必要的。另外，計算過程需要設定的數值或參數最好用變數來替代，並於程式（或模組）前頭加以定義，如此一來，再修改時才不會漏掉對應參數的重設定。一個不好的示範是，在 5.5 的程式中，重複出現了一些數字，這在修改時將容易產生錯誤。

最後，鼓勵大家花一些時間自己寫計量程式作業。這過程像小孩學說話般，一開始牙牙學語，但字彙與語法熟悉了，就開始說不停。不過，與一般口語溝通不同的是，寫程式是跟電腦說話，用字遣詞要十分精準。精準溝通的好處是，人與人之間不會產生誤會。假設 A 與 B 兩人研究 C 程式，一旦看懂後，A 與 B 對於 C 程式的解讀應該是相同的；反之，A 與 B 看一幅畫，也許一眼就看清楚畫中物體，但對於畫者欲表達的概念可能有各自的解讀。由於我們是讀博士班，若要做好研究，一個重要前提是精準溝通，這一點，我從管老師身上學到很多，而撰寫程式亦是有用的練習。在管老師的課堂上，大家將會慢慢地體認到此點。

練習題

1. Generate random samples with numbers of $T=50$, $T=100$, $T=300$, and $T=1000$ form the following distributions for 1000 times, and compute their sample averages each time. Plot the histograms under those cases as that in Subsection 4.2. Do your results obey the law of large numbers? Why?
(1) Chi-squared $\chi^2(1)$ distribution
(2) Student $t(5)$ with zero mean
(3) Student $t(1)$.

2. The ARMA(1,1) model is $y_t = \rho y_{t-1} + \varepsilon_t + \theta\varepsilon_{t-1}$. Simply let $y_1 = 100$ and then

generate the following cases under $\varepsilon_t \sim N(0,1)$ for $t = 1,2,3,\cdots,500$. Plot your results as the graph in Subsection 4.3. Which cases are likely as a history of a stock price?

(1) $(\rho,\theta) = (0,0)$
(2) $(\rho,\theta) = (1,0)$
(3) $(\rho,\theta) = (0,1)$
(4) $(\rho,\theta) = (1,1)$

3. Generate two sequences of $\{y_t\}$ and $\{x_t\}$ with $t = 1,2,...,T$ in the following settings, and then compute the t-statistic for β in the OLS of $y_t = \alpha + \beta x_t + \varepsilon_t$. Repeat this simulation 1000 times and count the number of cases in which their t-statistics are greater than the critical value of 1.96. Summarize you results for $T = 20,50,100,$ and 200, respectively.

(1) $y_t \sim N(0,1)$ and $x_t \sim N(0,1)$.
(2) $x_t \sim N(0,1)$ and $y_t = 0.8x_t + \eta_t$ with $\eta_t \sim N(0,1)$.
(3) $y_t = y_{t-1} + e_t$ with $e_t \sim N(0,1)$ and $x_t = x_{t-1} + \eta_t$ with $\eta_t \sim N(0,1)$.

4. The model of first-order autoregressive, AR(1), is specified as $y_t = \rho y_{t-1} + \varepsilon_t$. In this simulation, we simply let $y_1 = 0$ and consider four cases of $\rho = 0.1$, $\rho = 0.5$, $\rho = 0.9$, and $\rho = 0.99$ under $\varepsilon_t \sim N(0,1)$ for $t = 2,3,\cdots,T$. In each case, you should generate corresponding AR(1) sequences with numbers of $T = 50$, $T = 100$, $T = 300,$ and $T = 1000$ for 1000 times and compute their sample averages each time, and then plot the histograms of averages as those in Subsection 4.2. Please separately display and discuss your results under different cases.

5. In this simulation, random sequences $\{x_t\}$ with sizes of $T = 10$, $T = 50$, $T = 100$, and $T = 500$ are drawn from a distribution for 1000 times and their normalized sample averages are computed each time as follows:

$$\frac{\sqrt{T}(\bar{x} - \mu)}{\sigma},$$

where $\bar{x}$, μ, and σ are the sample average, mean, and standard deviation, respectively. Please plot the histograms of averages under different sizes of T and explain the results. Moreover, there are 3 distributions we should consider in this exercise as follows.

(1) Student t (2) distribution with zero mean.
(2) Student t (4) distribution with zero mean.
(3) Lognormal distribution.