統計計算與模擬

指導老師：余清祥 教授

姓名：106354020 許承恩

106354005 余佑駿

第一題

(a) Write a computer program using the Mid-Square Method using 6 digits to generate 10,000 random numbers ranging over [0, 999999]. Use the Kolmogorov-Smirnov Goodness-of-fit test to see if the random numbers that you create are uniformly distributed. (Note: You must notify the initial seed number used, and you may adapt 0.05 as the α value. Also, **you may find warning messages for conducting the Goodness-of-fit test, and comment on the Goodness-of-fit test**.)

此為$\mathrm{Mid-square\ \ Method}$產生亂數方式。在0~999999之間，產生10,000個亂數。利用Kolmogorov-Smirnov Goodness-of-fit test去檢驗亂數是否符合均勻分佈(uniformly distributed)。要說明initial seed number，並設$\alpha = 0.05$。並解釋為何在使用ks-test會有warning messages 。

(b) Consider the combination of 3 multiplicative congruential generators, i.e.,

$$u_i = \frac{x_i}{30269} + \frac{y_i}{30307} + \frac{z_i}{30323} (mod\ \ 1)$$

With $x_i = 171x_{i-1}(mod\ \ 30269), y_i = 172y_{i-1}(mod\ \ 30307), z_i = 170z_{i-1}(mod\ \ 30323)$ Use both the χ^2 and Kolmogorov-Smirnov Goodness-of-fit tests to check if the data are from $U(0,1)$ distribution.

此為LCG線性同餘法產生亂數，利用chi-square test and K-S test檢驗是否符合均勻分佈$U(0,1)$

(c) The calculators use $U_{n+1} = (\pi + U_n)^5 (mod\ \ 1)$ to generate random numbers between 0 and 1. Compare the result with those in (a) & (b), and discuss your finding based on the comparison. 此為同餘法。

利用$U_{n+1} = (\pi + U_n)^5 (mod\ \ 1)$去生成0~1之間的亂數，試比較(a),(b),(c)三種產生亂數的方法。

參考資料

https://www.cnblogs.com/arkenstone/p/5496761.html

http://archived.chns.org/s.php@page=11&id=34&id2=1222.html

Ks-test (Kolmogorov-Smirnov Goodness-of-fit test)

◎基本介紹與優缺點

Ks-test 做為雙樣本檢定，是用來比較「兩個觀測值分佈是否來自相同的連續型分配」的檢驗方法。

Ks-test 做為單樣本檢定，是用來比較「一個頻率分佈$f(x)$與理論分佈$g(x)$是否相同」的檢驗方法。

- ✓ 虛無假設H_0：兩個資料分佈一致 or 資料符合理論分佈
- ✓ 理論上 ks-test 只能用在連續型分配的檢定唷!!

Ks-test 實用的原因有兩個。第一，它是一種非參數檢定，所以不必事先考慮資料分配的假設；第二，它是用於所有的分配，以樣本為基礎，檢查潛在母體的分配位置、散布性(dispersion)，與圖形形狀，並比較樣本中的特徵是否具有一致性。

方便的代價就是當檢驗的資料分佈符合特定的分佈時，Ks-test 檢驗的靈敏度沒有相應的檢驗來的高。在樣本量比較小的時候，Ks-test 檢驗是非參數檢驗在分析兩組資料之間是否不同時，相當常用的方法。

◎使用方法

$ks.test(f, g, alternative = c(\text{two.sided}, \text{ less}, \text{ greater}), exact = NULL)$

f、g	要檢驗的兩個樣本，或是要檢驗的cdf與理論分布cdf
alternative	要單尾檢定還是雙尾檢定
exact	是否需要計算精確的P值

結果會顯示三個東西：$data$、D、$p-value$、$alternative\ \ hypothesis$

- ➢ $data$：比較的 data 為何。
- ➢ $D = max|f(x) - g(x)|$，兩個 cdf 的最大垂直差距為何
- ➢ $p - value$大於$\alpha = 0.05$，則不拒絕H_0。
- ➢ $alternative\ \ hypothesis$：使用單尾或雙尾檢定

當D很小，$p - value$大於$\alpha = 0.05$時，不拒絕H_0，推論數據來自某分布。

在做單樣本 K-S 檢驗或者正態檢驗時，有時會有錯誤提示“Kolmogorov – Smirnov 檢驗裡不應該有連結”(ties should not be present for the Kolmogorov-Smirnov test)，這是因為 Ks-test 檢驗只對連續cdf有效，而連續cdf中出現相同值的概率為 0，因此 R 會報錯。所以在做常態性檢驗前，要先對資料進行描述性分析，對資料整體要有大致的認識，後續才能選擇正確的檢驗方法。

Chi-square Test (χ $test$)

◎基本介紹與優缺點

卡方適合度檢定($Chi - Square\ \ Goodness\ \ of\ \ Fit\ \ Test$)，可以用來

1. 某組資料是否符合某個特定的機率分布？
2. 一組資料中幾個類別發生的機率，是否服從已知的比例關係？
   - ✓ 虛無假設H_0：該組符合特定的機率分布 or 資料服從已知的比例關係

◎使用方法

$chisq.test(x, p, rescale.p = FALSE)$

x	儲存各分類次數的向量
p	各分類的機率(預設值為各分類機率相等)
rescale.p	設定是否要轉換p選項中的機率值，使得所有機率值加總為1.0

結果會顯示三個東西：$data$、$X - squared$、df 、$p - value$

- ➢ $data$：要檢驗的 data，即各分類次數的值。
- ➢ $X - squared$，計算出的χ值。
- ➢ df：卡方的的自由度。
- ➢ $\mathrm{p - value}$：大於$\alpha = 0.05$，則不拒絕H_0。

當$p - value > \alpha = 0.05$，不拒絕$\mathrm{H_0}$，推論數據來特定的機率分布，或是服從已知的比例關係。

(a)小題解題流程

step01 設定$seed\ \ number = 122$

step02 先利用$runif(n = 1, min = 0, max = 1)$，由均勻分布生成一個亂數，並乘以 1000000，取整數部分，如此一來就會得到一個六位數，將此六位數訂為起始數(start)

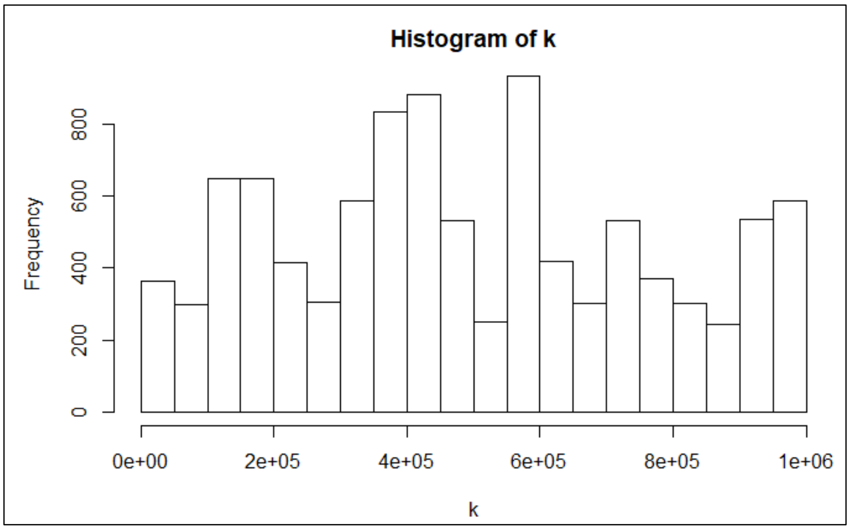
step03 之後$(start)^2$可得到一個 12 位數，接著我們取由左至右第 2~7 位的數字，得到一個六位數。重複此步驟 10000 次，共得到 10000 個數字。

step04 如果這是一個符合均勻分佈的亂數方式，那麼亂數的分布應該要很平均分散，稍微統計一下各種位數的數量，以及畫個直方圖做觀察。

step05 用 ks.test()檢驗該亂是使否為$uniformly\ \ distribution, U(0{,}999999)$

位數	4	5	6	總計
數量	4	657	9339	10000

由上表以及右圖，可以簡單猜測這種方法並不是一個好的均勻分佈亂數方式。



程式碼

```
set.seed(122)
start<-(1000000*runif(1))%>% floor() ;start
k<-c()
for(i in 1:10000){
  k[i]<-start^2 %>% substr(.,2,7) %>% as.numeric()
  start<-start^2 %>% substr(.,2,7) %>% as.numeric()
}
nchar(k) %>% table()
hist(k)
ks.test(k,"punif",0,999999) #punif 均勻分布累積機率值
```

結果與解釋

```
## Warning in ks.test(k, "punif", 0, 999999): ties should not be present
for the Kolmogorov-Smirnov test
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  k
## D = 0.072443, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

會出現 warning message 是因為 Ks-test 檢驗只對連續cdf有效，所以 R 會提醒妳這樣的檢驗不是很恰當，但 Ks-test 的結果依舊會跑出來。其中，樣本和均勻分布的累積密度函數，最大垂直差距$D = 0.072443$，計算出的$p - value < 2.2e - 16 < 0.05 = \alpha$，在雙尾檢定下，因此拒絕$\mathrm{H}_0$，樣本和均勻分配有所差異。

(b)小題解題流程

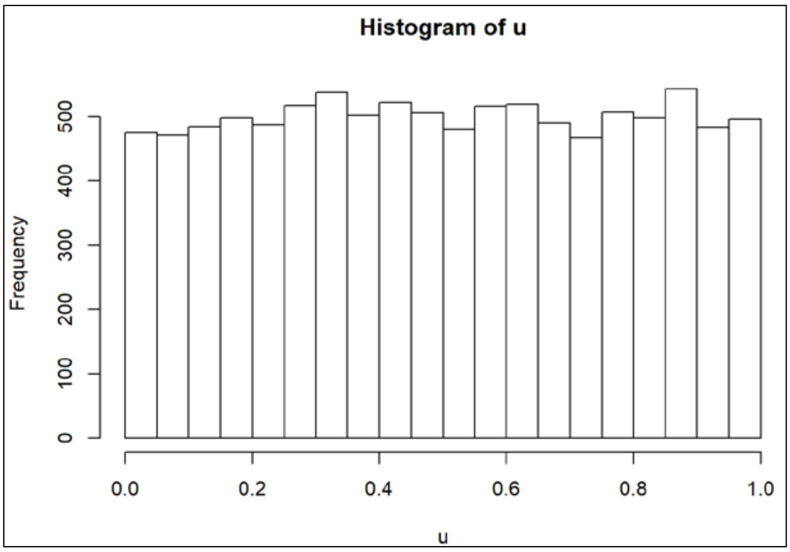
step01 設定(x, y, z)的初始值分別為$(1,1,1)$

step02 利用LCG線性同餘法，$u_i = \frac{x_i}{30269} + \frac{y_i}{30307} + \frac{z_i}{30323} (mod \ \ 1)$，產生 10000 筆數據。

step03 做個敘述性統計，並畫個直方圖做觀察。

step04 用 ks.test()檢驗該亂是使否為$uniformly \ \ distribution, U(0,1)$

由右圖看可以看出，這 10000 筆隨機生成的亂數，分布上蠻符合均勻分布的樣子。表一數據更顯示了每個分類的次數，大約都在 500 的上下。



表二為 10000 筆隨機亂數的敘述性統計，可以看出第一四分位數約為 0.26，中位數約為 0.5，第三四分位數約為 0.75；平均數約為 0.5。符合我們對均勻分布的猜想。

[表一 10000 筆亂數的直方圖每個分類的次數]

```
##  [1] 475 471 484 498 487 517 538 502 522 506 481 516 519 490 467 507 498
## [18] 543 483 496
```

[表二 10000 筆隨機亂數的敘述性統計]

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000256 0.2580882 0.4999813 0.5023883 0.7530791 0.9999387
```

因此，綜合直方圖、表一語表二數據，可知此亂數應該蠻符合均勻分布。

程式碼

```
x<-c(1);y<-c(1);z<-c(1)
u<-c()
for(i in 2:10001){
  x[i]<-(x[i-1]*171)%%30269
  y[i]<-(y[i-1]*172)%%30307
  z[i]<-(z[i-1]*170)%%30323
  u[i]<-((x[i]/30269)+(y[i]/30307)+(z[i]/30323))%%1
}
u<-u[-1]
summary(u); h<-hist(u);h$counts
ks.test(u,"punif")
chisq.test(h$counts)
```

結果與解釋

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  u
## D = 0.009332, p-value = 0.3486
## alternative hypothesis: two-sided
```

Ks-test 檢驗的結果顯示，該數據和均勻分布的累積密度函數，最大垂直差距$D = 0.009332$，計算出$p - value = 0.3486 > 0.05 = \alpha$，在雙尾檢定下，因此不拒絕$\mathrm{H}_0$，可推論該數據和均勻分配雷同。

```
##
##  Chi-squared test for given probabilities
##
## data:  h$counts
## X-squared = 17.052, df = 19, p-value = 0.5863
```

$Chi - squared\ \ test$檢驗的結果顯示，$\chi = 17.052$，$\alpha = 0.05$，自由度 19 的條件下，計算出$p - value = 0.3486 > \alpha$，因此不拒絕$\mathrm{H}_0$，可推論該數據和均勻分配雷同。

(c)小題解題流程

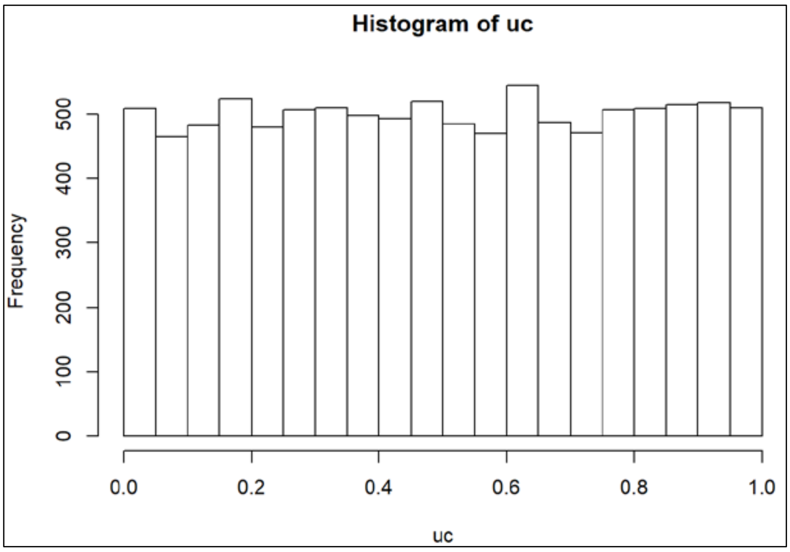
step01 設定初始值為 1

step02 利用同餘法，$U_{n+1} = (\pi + U_n)^5 (mod \ \ 1)$，產生 10000 筆數據。

step03 做個敘述性統計，並畫個直方圖做觀察。

step04 用 ks.test()和 chisq.test()檢驗該亂是使否為$uniformly \ \ distribution, U(0,1)$。並用 ks.test()檢驗此筆數據與(b)小題是否具有相同分配。

由右圖看可以看出，這 10000 筆隨機生成的亂數，分布上蠻符合均勻分布的樣子。表一數據更顯示了每個分類的次數，大約都在 500 的上下。



表二為 10000 筆隨機亂數的敘述性統計，可以看出第一四分位數約為 0.25，中位數約為 0.5，第三四分位數約為 0.75；平均數約為 0.5。符合我們對均勻分布的猜想。

[表一 10000 筆亂數的直方圖每個分類的次數]

```
##  [1] 509 465 482 524 480 507 510 498 493 519 484 470 544 487 471 507 509
## [18] 514 517 510
```

[表二 10000 筆隨機亂數的敘述性統計]

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0001451 0.2537806 0.5012201 0.5030918 0.7556807 0.9998672
```

程式碼

```
uc<-c(1)
for(i in 2:10001){
  uc[i]<-((uc[i-1]+pi)^5)%%1
}
uc <- uc[-1];huc <- hist(uc)
huc$counts;summary(uc)
ks.test(uc,"punif");chisq.test(huc$counts)
ks.test(uc,u)
```

結果與解釋

Ks-test：$\mathrm{p-value} = 0.6604 > 0.05$，不拒絕$H_0$，此 10000 筆數據和均勻分布雷同。

```
##  One-sample Kolmogorov-Smirnov test
##
## data:  uc
## D = 0.0073026, p-value = 0.6604
## alternative hypothesis: two-sided
```

Chi-square test：$\mathrm{p-value} = 0.6591 > 0.05$，不拒絕$H_0$，此 10000 筆數據和均勻分布雷同。

```
##  Chi-squared test for given probabilities
##
## data:  huc$counts
## X-squared = 15.972, df = 19, p-value = 0.6591
```

Ks-test：$\mathrm{p-value} = 0.9526 > 0.05$，不拒絕$H_0$，(b)和(c)的亂數有相似的分布。

```
##  Two-sample Kolmogorov-Smirnov test
##
## data:  uc and u
## D = 0.0073, p-value = 0.9526
## alternative hypothesis: two-sided
```

總結

(a)小題由直方圖可以看出，數字會出現規律，所以在 ks.test()檢驗下，和均勻分布相同的假設會被拒絕。(b)(c)小題則有類似的直方圖，且個別做 ks.test()及 chisq.test()檢驗都不拒絕，可推論和均勻分布相似。最後，對(b)(c)小題做 ks.test()檢驗，發現$p - value = 0.9526$，所以不拒絕虛無假設，(b)(c)小題所產生的亂數，具有相同的分布。

第二題

There are several ways for checking the goodness-of-fit for empirical data. In specific, there are a lot of normality tests available in R.

Generate a random sample of size 10, 50, and 100 from N(0,1) and t-distribution (with degrees 10 and 20) in R. You may treat testing random numbers from t-distribution as the power. For a level of significance α = 0.05 test, choose at least four normality tests in R (“nortest” module) to check if this sample is from $N(0,1)$. Tests used can include the Kolmogorov-Smirnov test and the Cramer-von Mises test. Note that you need to compare the differences among the tests you choose.

有許多種方法可以檢驗一筆數據的適合度。在 R 軟體中，有許多的常態性檢定，R 中有許多正態性測試。

利用 R 軟體，從$\mathrm{N}(0,1)$、$\mathrm{t}-$分布(自由度 10 和 20)，分別生成大小為 10、50、100 的隨機樣本。看看這些樣本是否能成功拒絕為$\mathrm{normal\ \ distribution}$的假設，做為檢定力的比較。設定顯著水準$\alpha = 0.05$，在 R 的 nortest package 中至少選擇四個常態性檢定，來檢查該樣本是否來自$\mathrm{N}(0,1)$。使用的測試可以包含 Kolmogorov-Smirnov test(ks-test)和 Cramer-von Mises test(cvm-test)。 最後，比較所選擇的檢定有甚麼差異。

解題思路

我們選定的常態性檢定函數為$\mathrm{ks.test}$、$\mathrm{cvm.test}$、$\mathrm{ad.test}$、$\mathrm{lillie.test}$

從$\mathrm{t}-$分配產生的隨機樣本，若拒絕為$\mathrm{normal\ \ distribution}$假設的成功次數越多，表示檢定力越強。至於拒絕的次數，我們將和把$\mathrm{N}(0,1)$產生的隨機樣本丟進相同的假設檢定下所被拒絕的次數做比較。

常態性檢定函數介紹

$\mathrm{ks.test}$：請見第一題。

$\mathrm{cvm.test}(x)$：x為隨機樣本，可以有缺失值存在，樣本扣除缺失值的大小必需> 7。

- W 值：W 越小，越接近 0，表示樣本資料越接近正態分佈
- p 值：如果$p - value$小於顯著性水準$\alpha = 0.05$，則拒絕H_0。(越大越好)

$\mathrm{ad.test}(x)$：x為隨機樣本，可以有缺失值存在，樣本扣除缺失值的大小必需> 7。

- A 值：A 越小，越接近 0，表示樣本資料越接近常態分佈
- p 值：如果$p - value$小於顯著性水準$\alpha = 0.05$，則拒絕H_0。(越大越好)

$\mathrm{lillie.test}(x)$：x為隨機樣本，可以有缺失值存在，樣本扣除缺失值的大小必需> 4。lillie.test 是基於 Kolmogorov–Smirnov test 的一種正態性檢驗。

- D 值：D 越小，越接近 0，表示樣本資料越接近正態分佈
- p 值：如果$p - value$小於顯著性水準$\alpha = 0.05$，則拒絕H_0。(越大越好)

程式流程

Step01 從$\mathrm{t} -$分配(自由度 10、20)分別隨機生成 10、50、100 筆數據，

Step02 從$\mathrm{N}(0,1)$隨機生成 10、50、100 筆數據

Step03 用$\mathrm{ks.test}$、$\mathrm{cvm.test}$、$\mathrm{ad.test}$、$\mathrm{lillie.test}$分別作常態性檢定

Step04 重複步驟 1~3，共 1000 次，看看$\mathrm{t} -$分配隨機生成的樣本被成功拒絕幾次，和$\mathrm{N}(0,1)$隨機生成樣本作檢定力比較。

結果說明

[$\mathbf{N = 10}$，模擬 1000 次]

三種隨機樣本在$\mathrm{ks.test}$、$\mathrm{cvm.test}$、$\mathrm{ad.test}$、$\mathrm{lillie.test}$常態檢定下被拒絕的次數。

N=10	ks.test	cvm.test	ad.test	lillie.test
T, df=10	64	64	78	68
T, df=20	51	57	47	53
Normal	41	53	37	51

以$Normal$被拒絕的次數當作基準點，可以發現，從$\mathrm{T} -$分配(自由度 10)生成的隨機樣本，都會被成功拒絕；而$\mathrm{T} -$分配(自由度 20)生成的隨機樣本，則被拒絕的次數和基準點差不多。

如果用 1000 次模擬被拒絕的比例去看，我們覺得$Normal$、$\mathrm{T} -$分配(自由度 20)都在 5%上下，差異不大；但明顯地$\mathrm{T} -$分配(自由度 10)則被拒絕的次數比較高(約 7~8%)，其中 ad.test 拒絕的效果最好(power 最強)。

[N = 50，模擬 1000 次]

三種隨機樣本在ks. test、cvm. test、ad. test、lillie. test常態檢定下被拒絕的次數。

N=50	ks.test	cvm.test	ad.test	lillie.test
T, df=10	46	117	128	100
T, df=20	46	58	57	55
Normal	56	42	38	49

以$Normal$被拒絕的次數當作基準點，可以發現，從$T-$分配(自由度 10)生成的隨機樣本，除了 ks.test 之外，都會被成功拒絕；而$T-$分配(自由度 20)生成的隨機樣本，則被拒絕的次數和基準點差不多。

如果用 1000 次模擬被拒絕的比例去看，我們覺得$Normal$、$T-$分配(自由度 20)都在 5%上下，差異不大；除了 ks.test 之外，$T-$分配(自由度 10)則被拒絕的次數比較高(約 10%)。所以由此可知，ks.test 的檢定力不好，ad.test 的效果最好，cvm.test 和 lillie.test 的效果則差不多。

[N = 100，模擬 1000 次]

三種隨機樣本在ks. test、cvm. test、ad. test、lillie. test常態檢定下被拒絕的次數。

N=100	ks.test	cvm.test	ad.test	lillie.test
T, df=10	54	149	165	137
T, df=20	58	76	78	67
Normal	48	52	45	58

以$Normal$被拒絕的次數當作基準點，可以發現，從$T-$分配(自由度 10)生成的隨機樣本，除了 ks.test 之外，都會被成功拒絕；而$T-$分配(自由度 20)生成的隨機樣本，則被拒絕的次數略大於基準點。

如果用 1000 次模擬被拒絕的比例去看，我們覺得這次$T-$分配(自由度 20)比$Normal$拒絕的次數略多，是因為N的大小比之前兩次都大，因此檢定力增加，但同樣地， ks.test 的檢定力依舊不高，不論是$T-$分配(自由度 10)或是$T-$分配(自由度 20)表現都不好。至於其餘的三種常態性檢定，ad.test 的檢定力最好，cvm.test 次之，lillie 最後，但仍遠比 ks.test 有效。

$T-$分配(自由度 10)則被拒絕的次數比較高(約 10%)。所以由此可知，ks.test 的檢定力不好，ad.test 的效果最好，cvm.test 和 lillie.test 的效果則差不多。

結論

如果要做常態性檢定，建議使用 ad.test。

程式碼

```r
test<-function(k,j){
  N<-c(10,50,100)
  D<-c(10,20)
  a<-c();b<-c();c<-c();d<-c()
  for(i in 1:1000){
    tmp1<-rt(N[k],D[j]) %>% ks.test(.,"pnorm") %>% .[2] %>%`<`(0.05) %>%
`*`(1) %>% as.numeric()
    tmp2<-rt(N[k],D[j]) %>% cvm.test() %>% .[2] %>%`<`(0.05) %>% `*`(1)
%>% as.numeric()
    tmp3<-rt(N[k],D[j]) %>% ad.test()  %>% .[2] %>%`<`(0.05) %>% `*`(1)
%>% as.numeric()
    tmp4<-rt(N[k],D[j]) %>% lillie.test() %>% .[2] %>% `<`(0.05) %>%
`*`(1) %>% as.numeric()
    a<-sum(a,tmp1);b<-sum(b,tmp2);c<-sum(c,tmp3);d<-sum(d,tmp4)
  }
  paste(a,b,c,d,sep = "/")
}

norr<-function(k){
  N<-c(10,50,100)
  a<-c();b<-c();c<-c();d<-c()
  for(i in 1:1000){
    tmp1<-rnorm(N[k]) %>% ks.test(.,"pnorm") %>% .[2] %>%`<`(0.05) %>%
`*`(1) %>% as.numeric()
    tmp2<-rnorm(N[k]) %>% cvm.test() %>% .[2] %>%`<`(0.05) %>% `*`(1) %>%
as.numeric()
    tmp3<-rnorm(N[k]) %>% ad.test()  %>% .[2] %>%`<`(0.05) %>% `*`(1) %>%
as.numeric()
    tmp4<-rnorm(N[k]) %>% lillie.test() %>% .[2] %>% `<`(0.05) %>% `*`(1)
%>% as.numeric()
    a<-sum(a,tmp1);b<-sum(b,tmp2);c<-sum(c,tmp3);d<-sum(d,tmp4)
  }
  paste(a,b,c,d,sep = "/")
}
```

第三題

Write your own R programs to perform **Gap test, Permutation test, and run test.** Then use this program to test if the uniform random numbers generated from Minitab (or SAS, SPSS, Excel) and R are independent

利用 R 語言設計 Gap test, Permutation test 以及 run test 的程式，接著用 R 和 Minitab (or SAS, SPSS, Excel)產生$uniform\ \ distribution$的亂數，並利用自己設計的三個程式去檢驗不同軟體跑出的亂數是否獨立(也就是個別軟體的亂數是否獨立產生)。

以下分別介紹 Gap test, Permutation test,和 run test

Gap test

選取介於 0, 1 間的兩個數 $0 < \alpha < \beta < 1$，再考慮介於α與β間的亂數。

介於α與β間的亂數間隔應與幾何分配有關，間隔個數

$$P(K = k) = (\beta - \alpha) \times \left(1 - (\beta - \alpha)\right)^k, k = 0,1, \dots$$

再套入卡方檢定。

Permutation test(置換檢驗)

Permutation Tests，屬於統計顯著性檢驗的一種。顯著性檢驗通常可以告訴我們一個觀測值是否是有效的，例如：檢驗兩組樣本均值差異的假設，可以告訴我們這兩組樣本的均值是否相等（或者哪個均值更大）。

Permutation Tests 檢驗下，虛無假設是假定二組沒有差別，由此將二組樣本合併，從中以無放回方式進行抽樣，分別歸入兩個組再計算統計量，反復進行由此得到置換分佈（即構造一個新的分佈），在此基礎上進行顯著性檢驗推斷是否拒絕虛無假設。

例如：如果要判斷兩個樣本集是否來源於同一分佈，那麼原假設就是二者來自於同一分佈，所以如果將兩個樣本集進行順序上的置換，多次重新做無放回抽樣後，對分佈的參數統計量的估計與置換前的統計量估計應保持大致相同（用 P 值來判斷）。

- ✓ Permutation Tests 是一種不同於 Bootstrap 的重採樣方法，不同之處在於 **Permutation Tests 再抽樣過程中是無放回抽樣。**
- ✓ 特別適用於分佈未知的小樣本資料，以及某些難以用常規方法分析資料的假設檢驗問題。

Run test(Up-and-Down test)

Run test 又稱為 Up-and-Down test，是用在研究一數列的事件，其中此數列中的每一個元素，都能被假設成事件的結果為成功或失敗其中之一。此檢定方法可檢定出，此事件的結果，成功或失敗是否隨機出現。

一數列中，相同元素的最大子數列，即稱為一個 run。run 的個數太多或太少，都代表著成功與失敗的出現並不隨機。令$U = run$的個數，$N =$ 數列的大小，則統計量$z = \frac{U-\frac{(2N-1)}{3}}{\sqrt{\frac{16N-29}{90}}} \sim N(0,1)$

檢定時，runs 的個數太多或太少皆顯示其不隨機性，所以我們將考慮雙邊檢定。虛無假設為成功與失敗隨機出現。即序列應參雜排列，且有許多 runs。

✓ 組數較多的時候不適用!!

解題思路過程

gap test 的操作過程

Step01 先設定一區間(α, β)

Step02 將隨機生成的亂數在區間內的設為 1，不在區間內的設為 0

Step03 計算相鄰 1 的間隔，並對其做卡方檢定

permutation test 的操作過程

Step01 將產生的亂數依先後順序每k個一組$(x_1, x_2, \ldots, x_k)(k_{k+1}, x_{k+2}, \ldots), \ldots$

Step02 每組再依大小順序寫出類似$(1,2, \ldots, k)$的格式，共有$k!$種可能。

Step03 計算每一種可能的個數，再以卡方檢定，檢查每一種可能發生率是否相同。

run test 的操作過程

Step01 比較連續兩個亂數間的大小，以正號(+或 1)表示遞增、負號(–或 0)表示遞減。

Step02 連續的 0 及 1 視為一個單位，稱為連(Run)

Step03 連(Run)總共的個數為U，期望值為$\frac{(2N-1)}{3}$，變異數為$\frac{16N-29}{90}$，則$z = \frac{U-\frac{(2N-1)}{3}}{\sqrt{\frac{16N-29}{90}}} \sim N(0,1)$，雙尾檢定，計算$p - value$值

程式碼

Gap test

```
gap.test=function(data,a,b){
  x<-ifelse(a<data & data<b,1,0)
   x1=which(x==1)
    y=x1[-1]-x1[-length(x1)]-1
     k<-unique(y) %>% sort() #看有哪些數字
      obs<-table(y) %>% as.numeric()# 實際值
       prob<-(b-a)*((1-b+a))^k #理論機率
      exp<-length(data)*prob
     stop<-which(exp<5) #從 stop 開始期望次數會小於 5
    go<-!(1:length(exp)%in%stop)
   obs<-c(obs[go],sum(obs[stop]))
  prob<-c(prob[go],1-sum(prob[go]))
 chisq.test(obs,p=prob)
}
```

Permutation test

```
permute.test=function(data,k){
  y=rep(10,k)^c((k-1):0)
  x=matrix(data,ncol=k,byrow=T)
  x1=apply(x,1,rank)
  yy=apply(x1*y,2,sum)
  stat<-table(yy) %>% as.numeric()
  chisq.test(stat)
}
```

Run test

```
run<-function(r){
  r1 <- 1*(r[-1]>r[-10000] ) #變號位置
sum(r1[-1] != r1[-9999]) + 1 # 共有幾個 run
r2 <- which(r1[-1] != r1[-9999]) #run 的位置
r3 <- table(r2[-1]-r2[-length(r2)]) #看 run＝n 的個數
r3
Z <- (sum(r3)-(2*10000-1)/3)/((16*10000-29)/90)^0.5
1-pnorm(Z)
```

```
}
```

檢定結果

從 R 及 Excel 隨機生成兩筆樣本，分別以Gap. test、Permutation. test、Run. test檢定獨立性

	Gap.test	Per.test	Run.test
R	0.1663	0.09301	0.8883096
Excel	0.5307	0.5121	0.939388

由上表的三種檢定可知，Excel 的亂數結果似乎比 R 的隨機樣本，獨立性來的更好。

第四題

$(\sum_{i=1}^{12} U_i - 6)$ can be used to approximate $N(0,1)$ distribution, where $Ui's$ are random sample from $U(0,1)$.

(a) Based on α = 0.05, compare the results of the Chi-square test and the Kolmogorov-Smirnov test, and see if there are any differences.

(b) Design two tests of independence (which are not the same as you saw in class) and apply them on the random sample that you generate.

利用$(\sum_{i=1}^{12} U_i - 6)$生成一組$N(0,1)$的隨機樣本， U_i是從$uniform\ \ distribution$中生成的亂數。

(a) 在顯著水準為$\alpha = 0.05$下，比較 Chi-square test、ks.test，看看有沒有不同。

(b) 設計兩個獨立性的檢定方法(不能和在課堂上看到的一樣)，並對生成的隨機樣本做檢定。

解題思路

(a)

Step01 利用$\sum_{i=1}^{12} U_i - 6$生成 1000 筆數據

Step02 在顯著水準為$\alpha = 0.05$下，比較 Chi-square test、ks.test 的結果

(b)

[方法一 進階版的 run test]

Step01 生成 1000 筆數據，將隨機生成的順序紀錄為 num

Step02 對 num 做小到大的排列，紀錄排列後的順序為 rank

Step03 計算生成順序的 1 號到排列順序的 1 號，距離幾個數字，並記錄為 run_num

Step04 最後對這 1000 個 run_num 做 run.test

[方法二 進階版的 permutation test]

Step01 生成 9999 筆數據，依照資料生成的順序給予 order 1,2,3,4,5,……紀錄為 num

Step02 對 num 由小排到大 從最大的開始給予編號 1,2,3,1,2,3,⋯,1,2,3,⋯，所以會有一千組的 1、2、3

Step03 觀察每一個 order 所對應的 rank 是否有規律 在這裡以三個為一組，將會有 3^3=27 種組合

Step04 以卡方鑑定測試 27 種組合出現機率是否相同

程式與結果解釋(a)

```
k<-c()
for(i in 1:10000){
  k[i]<-runif(12) %>% sum() %>% `-`(6)
}
ks.test(k,"pnorm")
```

```
nor<-rnorm(10000)
y<-cut(nor,breaks = c(-100,-2,-1,0,1,2,100)) %>% table() %>% `/`(10000)
x<-cut(k,breaks = c(-100,-2,-1,0,1,2,100)) %>% table() %>% `/`(10000)
p<-y %>% as.numeric()
chisq.test(x,p = p)
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  k
## D = 0.0070014, p-value = 0.711
## alternative hypothesis: two-sided
```

```
##
##  Chi-squared test for given probabilities
##
## data:  x
## X-squared = 0.0029136, df = 5, p-value = 1
```

所以，不論是 ks.test 或是卡方檢定，都不拒絕虛無假設，接受這組隨機生成的數據符合常態分布。

程式與結果解釋(b)

[第一種方法]

```
run<-function(r){
  r1 <- 1*(r[-1]>r[-1000] ) #變號位置
  sum(r1[-1] != r1[-999]) + 1 # 共有幾個run
  r2 <- which(r1[-1] != r1[-999]) #run 的位置
  r3 <- table(r2[-1]-r2[-length(r2)]) #看run=n 的個數
  r3
  Z <- (sum(r3)-(2*1000-1)/3)/((16*1000-29)/90)^0.5
  1-pnorm(Z)
}
GOGO <- function(){
  r <- runif(12, min = 0,max = 1)
  return(sum(r)-6)
}
A <- c();B <- c()
num <-c()
for(i in 1:1000){
  num[i] <- GOGO()
}
rank <- sort(num)
run_num <- c()
for(i in 1:1000){
  for(j in 1:1000){
    if(num[i]==rank[j]){
      run_num[i]=abs(i-j)+1
    }
  }
}
run(run_num)
```

為了看這個方法好不好，分別丟入 R 生成的亂數、第一題模擬的亂數(mid-square)以及本題生成的亂數做比較。

	R生成亂數	Mid-Square亂數	本題亂數
p-value	0.8628	< 2.2e-16	0.889922

我們可以發現，成功拒絕了 Mid-Square 的亂數，而本題生成的亂數與 R 生成的亂數則不拒絕，所以這個方法是有效的。

[第二種方法]

```
newp<-function(ram){
  num1<-sort(ram,decreasing = TRUE)
   dff<-data.frame(num1=ram,order=1:9999)
    df<-data.frame(rank=rep(c(1,2,3),3333),num1=num1)
     final<-merge(df,dff,by="num1")
      newtable<-final[order(final$order),] %>% .[,2]

      mx=matrix(newtable,ncol=3,byrow=T)
     my<-matrix(c(1,10,100),ncol = 1)
    stat<-mx%*%my %>% .[,1] %>% table() %>% as.numeric #test statistic
   chisq.test(stat)
}
```

為了看這個方法好不好，分別丟入 R 生成的亂數、第一題模擬的亂數(mid-square)以及本題生成的亂數做比較。

	R生成亂數	Mid-Square亂數	本題亂數
p-value	0.8628	< 2.2e-16	0.7939

我們可以發現，成功拒絕了 Mid-Square 的亂數，而本題生成的亂數與 R 生成的亂數則不拒絕，所以這個方法也是有效的。

第五題

Use the bisection, false positions, and/or secant methods to find the roots, and check your answers with the functions in R (e.g., "uniroot").

You need to specify the **starting points**, **convergence criterion**, and **number of iterations**.

(a) $f(x) = e - \frac{1}{3.5+x}$ (b) $f(x) = \frac{e^x}{\sqrt{1+x^2}} - 0.5$

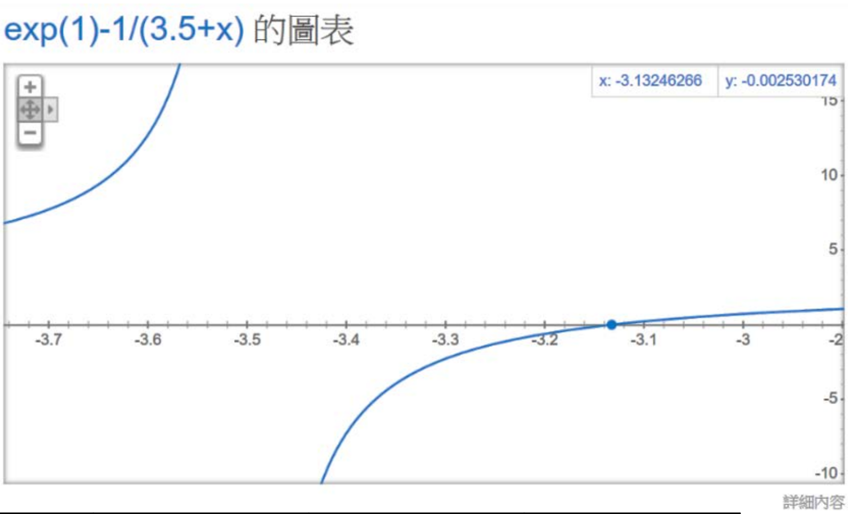
(c) $the\ \ eigen-values\ \ of\ \ matrix\ \ A = \begin{bmatrix} 2 & 3 & 4 \\ 3 & 4 & 5 \\ 4 & 5 & 6 \end{bmatrix}$

請分別用 bisection(二分法)、false position(假位法)、secant method(割線法)，對下面的 $(a),(b),(c)$小題求根。利用 R 的 stats package 中的"uniroot" function 確認答案。要特別標記求根時的 starting point(起始位置), convergence criterion(收斂準則), number of iterations(疊代次數)。

結果說明

(a) $f(x) = e - \frac{1}{3.5+x}$

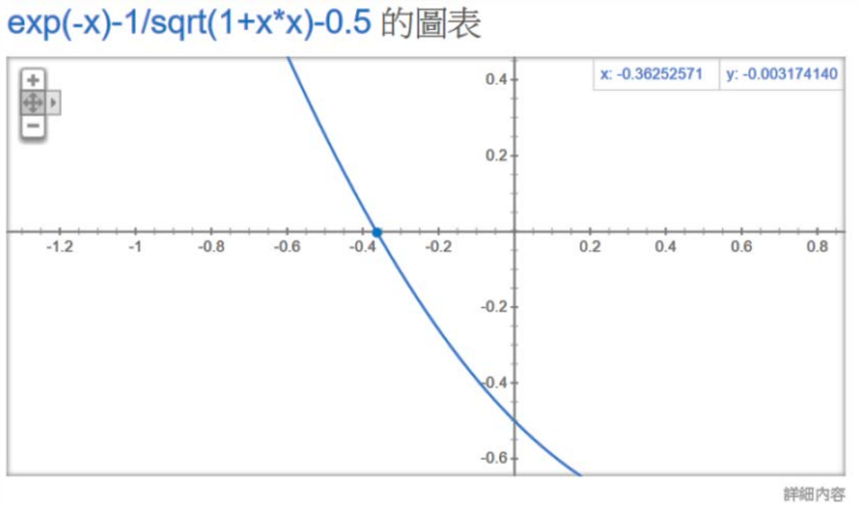
starting points	(-3.5,-2)
convergence criterion	1.00E-06



Method	bisection	false position	secant method	uniroot
number of iterations	22	6	6	4
root	-3.13212037086	-3.13212055883	-3.13212055884	-3.13211452412

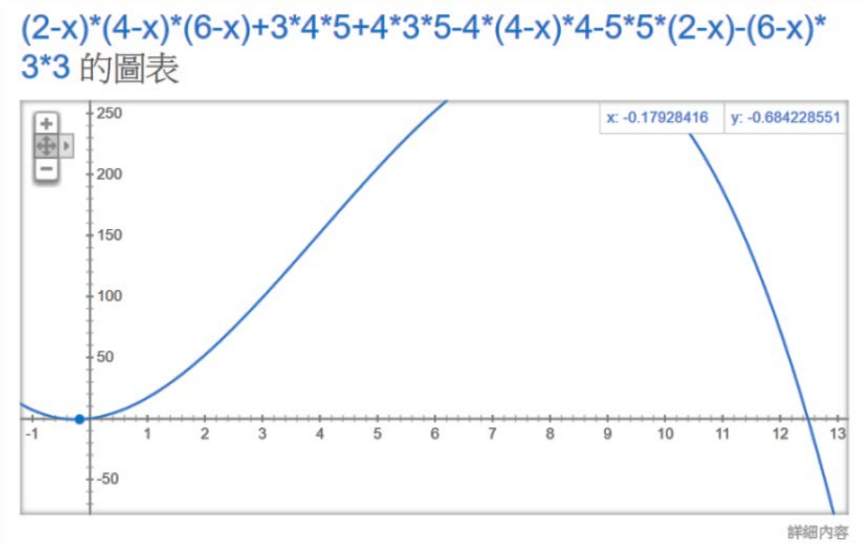
(b) $f(x) = \frac{e^x}{\sqrt{1+x^2}} - 0.5$

starting points	(-3,3)
convergence criterion	1.00E-06



Method	bisection	false position	secant method	uniroot
number of iterations	49	7	9	8
root	0.55772869892	0.55772869898	0.55772869898	0.55773748227

(c) $the\ \ eigen-values\ \ of\ \ matrix\ \ A = \begin{bmatrix} 2 & 3 & 4 \\ 3 & 4 & 5 \\ 4 & 5 & 6 \end{bmatrix}$



starting points	(-0.6,-0.3)(-0.3,0.2)(12,14)
convergence criterion	1.00E-06

Method	bisection	false position	secant method	uniroot
number of iterations	19	11	7	6
root One	-0.48073998	-0.48074050	-0.48074472	-0.48074099
number of iterations	19	12	11	6
root Two	0.0000011	-0.0000002	0.0000000	0.0000008
number of iterations	21	9	6	8
root Three	12.4807415	12.4807407	12.4807396	12.4807367

結論

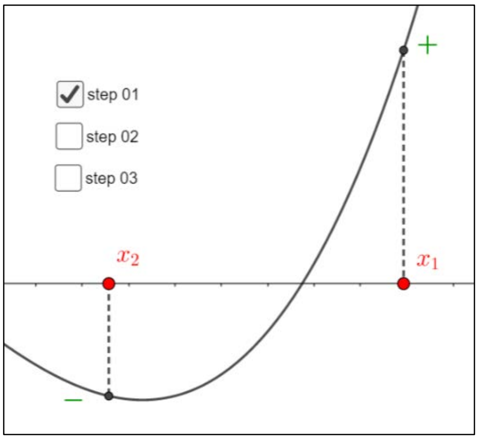
由運行結果可知，Bisection Method 找根的效率是最差的，雖然 False Position 運用了類似勘根的概念，但是他在收斂的速度會比 Bisection Method 快上許多。

而 False Position、Secant Method 以及 uniroot 的效率差不多

以下先分別介紹 bisection(二分法)、false position(假位法)、secant method(割線法)的原理。

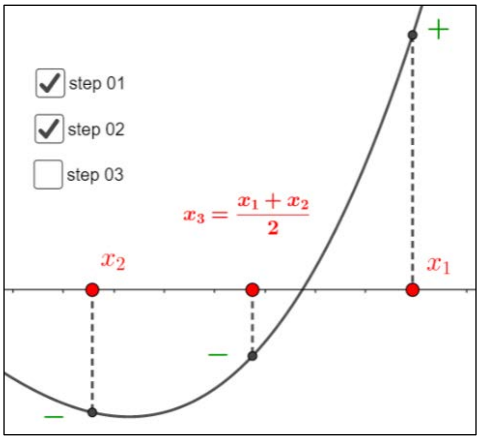
Bisection Method(二分法)

類似勘根定裡的概念。



step 01：

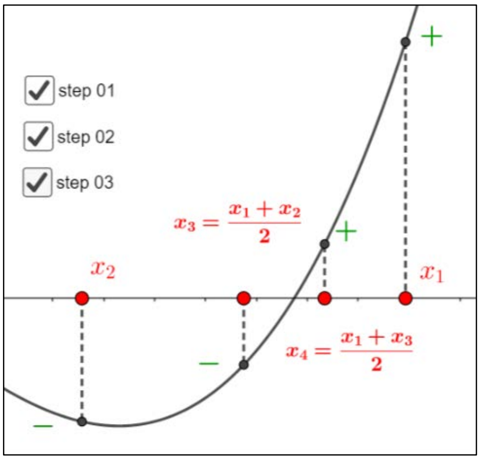
在根的兩次先找兩個起始點x_1, x_2，函數值相乘為異號(即一正一負，$f(x_1) \times f(x_2) < 0$)



step 02：

令 $x_3 = \frac{x_1+x_2}{2}$ ，判斷$f(x_3)$的正負值，找出$f(x_1)$和$f(x_2)$哪個和$f(x_3)$相乘為異號，則方程式的根就在兩點之間。

在右圖中因為 $f(x_1) \times f(x_3) < 0$，所以可知方程式的根在(x_1, x_3)之間。



step 03：

因為方程式的根在(x_1, x_3)之間，所以令 $x_4 = \frac{x_1+x_4}{2}$ 接著判斷$f(x_4)$的正負值，$f(x_1)$和$f(x_3)$哪個和$f(x_4)$相乘為異號，則方程式的根就在兩點之間。

在右圖中因為 $f(x_3) \times f(x_4) < 0$，所以可知方程式的根在(x_3, x_4)之間。

由 step 01,02,03 可知，透過判斷判斷函數值相乘為異號的方式，可以使我們取中點(二分法)後，新產生的x坐標離方程式的根越來越近，因此可以得到方程式根的近似值。

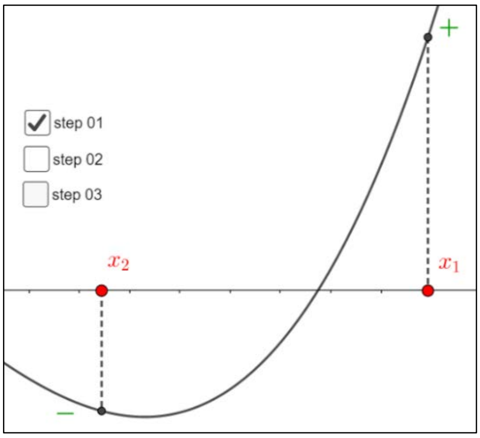
值得注意的是，要使用 Bisection Method(二分法)，前提是函數必須是連續的，或者是在使用的範圍內為連續圖形，否則會求不到根！

→優點：原理簡單，利用勘根定理。且前提只要使用範圍的函數圖形為連續即可。

→缺點：尋找時間冗長。無法求複數根。且必須使用到 f(a)和 f(b)的資訊。

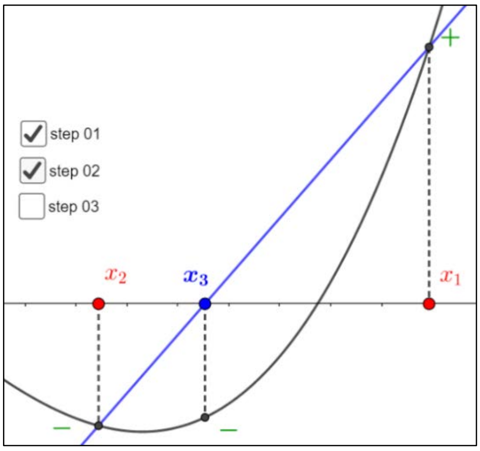
false position

又稱為 Regular Falsi method，依舊類似勘根定裡的概念。



step 01：

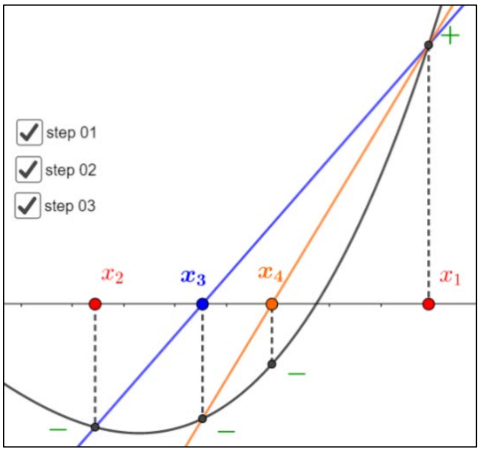
在根的兩次先找兩個起始點x_1, x_2，函數值相乘為異號(即一正一負，$f(x_1) \times f(x_2) < 0$)



step 02：

將$f(x_1)$和$f(x_2)$連線，交$x-axis$於x_3，判斷$f(x_3)$的正負值，找出$f(x_1)$和$f(x_2)$哪個和$f(x_3)$相乘為異號，則方程式的根就在兩點之間。

在右圖中因為 $f(x_1) \times f(x_3) < 0$，所以可知方程式的根在(x_1, x_3)之間。



step 03：

因為方程式的根在(x_1, x_3)之間，所以將$f(x_1)$和$f(x_3)$連線，交$x-axis$於x_4 接著判斷$f(x_4)$的正負值，$f(x_1)$和$f(x_3)$哪個和$f(x_4)$相乘為異號，則方程式的根就在兩點之間。

在右圖中因為 $f(x_1) \times f(x_4) < 0$，所以可知方程式的根在(x_1, x_4)之間。

由 step 01,02,03 可知，透過判斷判斷函數值相乘為異號的方式，可以使我們新產生的x坐標離方程式的根越來越近，因此可以得到方程式根的近似值。

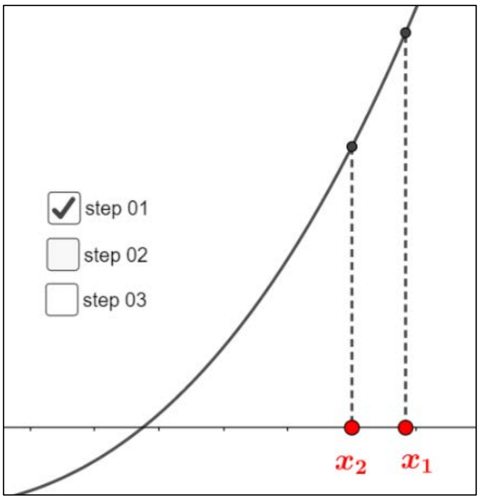
一樣值得注意的是，要使用 False Position Method，前提是函數必須是連續的，或者是在使用的範圍內為連續圖形，否則會求不到根！

→優點：比起二分法，在判斷時多使用了(a,f(a))和(b,f(b))的資訊

→缺點：依舊透過勘根定理尋找根的範圍，尋找時間冗長。

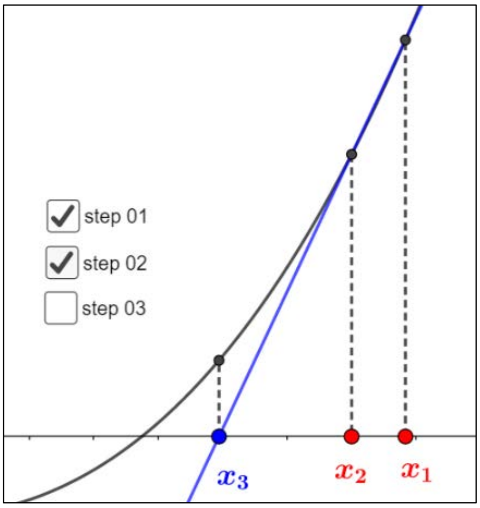
secant method(割線法)

依舊類似勘根定裡的概念。



step 01：

secant method 一開始的兩點不必一定要在根的兩側，在同側即可。如右圖，x_1, x_2在方程式根的同側。

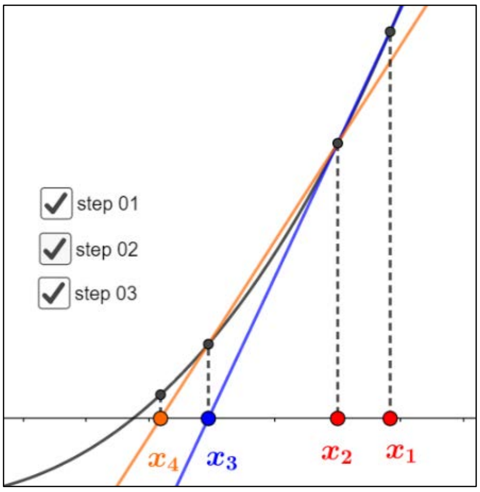


step 02：

將$f(x_1)$和$f(x_2)$連線，交$x-axis$於x_3。

step 03：

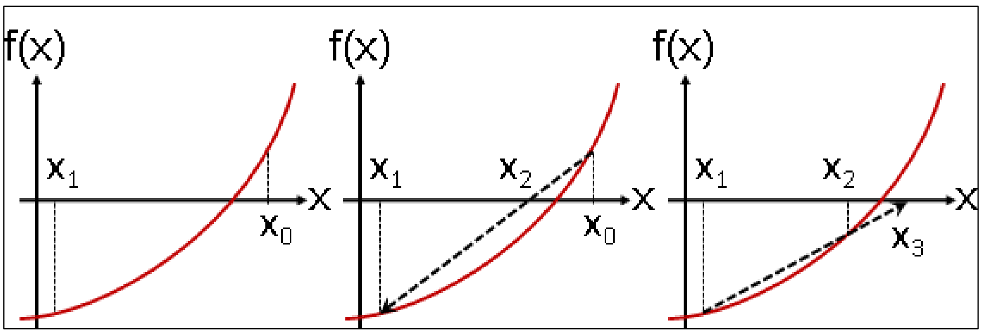
和 step 02 類似，將$f(x_2)$和$f(x_3)$連線，交$x-axis$於x_4。如此一來，新生成的點便離方程式的根越來越近。



由 step 01,02,03 可知，secant method 比較著重在遞迴的概念，新生成的x坐標的方式是由最新的兩個點，對應上去的函數值做割線交$x-axis$產生的，新生成的點會離方程式的根越來越近，因此可以得到方程式根的近似值。

p.s.即使是初始的兩點在異側也沒問題。

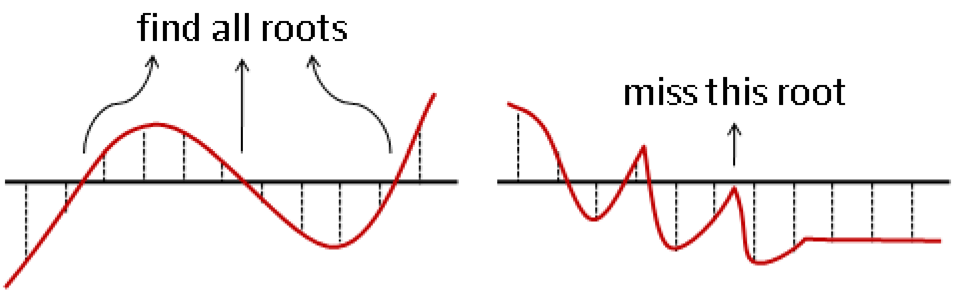
右圖為一開始的起始點在方程式根的異側情形。



Bisection Method 使用上的隱憂

整條數線細分成許多微小區間。 $f(a)$和$f(b)$同號的區間，視作沒有根；$f(a)$和$f(b)$為零的區間，視作只有端點有根；$f(a)$和$f(b)$異號的區間，視作剛好有一個根，實施二分法找到根。

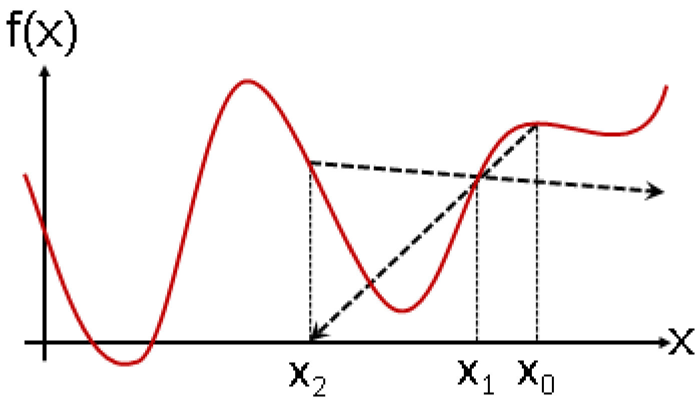
只要細分的足夠細膩，理論上可以找到所有根，除了一種例外：根恰好是區域極值。此時必須配合其他的求根方法，才能處理這個例外。



Secant Method 使用上的隱憂

一開始選定的兩點，如果不夠靠近根，割線可能會亂跑。

運氣不好時，割線呈水平線，割線法會故障；割線趨近水平線，下一點會溢位。



參考資料：

http://www.csie.ntnu.edu.tw/~u91029/RootFinding.html#2

R 函數 uniroot 參數介紹

$uniroot(f, interval, \ldots, tol =, maxiter = 1000)$

f	the function for which the root is sought. 要找根的方程式。
Interval	a vector containing the end-points of the interval to be searched for the root. 要讓R程式尋找根的一個閉區間範圍
tol	the desired accuracy (convergence tolerance). 精確度(可接受的誤差值)
maxiter	the maximum number of iterations. 疊代次數。

程式碼

[Bisection Method]

```
while(right-left>1e-6){
  m<-sum(left,right)/2
  ifelse(fcn(left)*fcn(m)<0,right<-m,left<-m)
  i<-i+1
}
paste0("The root is ",m,"\ and i = ",i)
```

[Fasle Position Method]

```
while(abs(a-b)>1e-6|(fcn(a)*fcn(b)>0)){
  m1<-(fcn(a)-fcn(b))/(a-b)
   b1<-fcn(a)-m1*a
    tmp1<-(-(b1/m1))
    m2<-(fcn(tmp1)-fcn(a))/(tmp1-a)
   b2<-fcn(tmp1)-m2*tmp1
  tmp2<-(-(b2/m2))
 a<-tmp2;b<-tmp1
 i<-i+1
}
paste0("The root is ",tmp2,"\ and i = ",i)
```

[Secant Method]

```
while(abs(a-b)>1e-6){
m1<-(fcn(a)-fcn(b))/(a-b)
b1<-fcn(a)-m1*a;tmp1<-(-(b1/m1));m2<-(fcn(tmp1)-fcn(b))/(tmp1-b)
b2<-fcn(tmp1)-m2*tmp1;tmp2<-(-(b2/m2));a<-tmp2;b<-tmp1
i<-i+1
}
paste0("The root is ",tmp2,"\ and i = ",i)
```

第六題

Consider a multinomial observation $X=(x_1,x_2,x_3,x_4)$ with class probabilities given by $(p_1,p_2,p_3,p_4)=\left(\frac{2+\theta}{4},\frac{1-\theta}{4},\frac{1-\theta}{4},\frac{\theta}{4}\right)$, where $0<\theta<1$. The sample size is $n=\sum x_i$ and the parameter θ 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).

解題思路

$$f(x|\theta)=\left(\frac{3839!}{1997!906!904!32!}\right)\cdot\left(\frac{2+\theta}{4}\right)^{1997}\cdot\left(\frac{1-\theta}{4}\right)^{906}\cdot\left(\frac{1-\theta}{4}\right)^{904}\cdot\left(\frac{\theta}{4}\right)^{32}$$

找$f(x|\theta)$的最大值，即為找$\frac{d}{d\theta}\log(f(x|\theta))=0$的解。

程式結果

Method	secant method	牛頓法
starting points	(a,b)為兩個隨機生成的兩個亂數	隨機生成的一個數
convergence criterion	1.00E-10	1.00E-06
root	0.03571230224	0.035712302

所以當$\hat{\theta}\approx 0.0357$的時候，可以使$x$的最大概似估計函數達到最大值。

程式碼

[Secant method]

```
fcn<-function(x){#這部分偷懶直接算
  1997/(2+x)-1810/(1-x)+32/x
}

secant<-function(){
a<-runif(1);b<-runif(1)
i<-1
while(abs(a-b)>1e-10){
m1<-(fcn(a)-fcn(b))/(a-b)
b1<-fcn(a)-m1*a
tmp1<-(-(b1/m1))

m2<-(fcn(tmp1)-fcn(b))/(tmp1-b)
b2<-fcn(tmp1)-m2*tmp1
tmp2<-(-(b2/m2))
a<-tmp2;b<-tmp1
a<-ifelse((a-b)=="NaN",0,a)
b<-ifelse((a-b)=="NaN",0,b)
i<-i+1
}
return(c(a,i))
# paste0("The root is ",a,"\ and i = ",i)
}
```

[(Another method)牛頓法]

```
intgo<-function(){
  r<-c(1,2)
  k<-2
  int<-runif(1)
  plus<-0.0000001
  while(abs(r[k]-r[k-1])>1e-6){#T 繼續做 F 停止
    x1<-c(int,fcn(int))
    x2<-c((int+plus),fcn(int+plus))
    m<-(x2[2]-x1[2])/plus
    b<-x2[2]-m*x2[1]
    int<-(-b/m)
    r[k+1]<-int
    k<-k+1
  }
  return(c(int,length(r)))
}
```

詳細程式碼請前往 http://rpubs.com/En9515/372403

參考資料

[chisq.test ref]

http://biostatdept.cmu.edu.tw/doc/epaper_a/paper/teaching_corner_051-1.pdf

http://chenyuren.blogspot.tw/2012/06/go-odness-of-fit-test-x-16-tablex-chisq.html

https://www.yiibai.com/r/r_chi_square_tests.html

[ks.test ref]

http://blog.sina.com.cn/s/blog_403aa80a01019ly5.html

https://www.cnblogs.com/arkenstone/p/5496761.html

http://blog.sciencenet.cn/blog-563898-877003.html

http://www.17bigdata.com/r%E8%AF%AD%E8%A8%80%E4%B8%8E%E6%AD%A3%E6%80%81%E6%80%A7%E6%A3%80%E9%AA%8C.html

http://www.17bigdata.com/r%E8%AF%AD%E8%A8%80%E4%B8%8E%E6%AD%A3%E6%80%81%E6%80%A7%E6%A3%80%E9%AA%8C.html

https://tw.answers.yahoo.com/question/index?qid=20110818000015KK08641

https://www.ibm.com/support/knowledgecenter/zh-tw/SSLVMB_24.0.0/spss/base/idh_ntk1.html

http://www1.pu.edu.tw/~tfchen/design_fs/C4_Nonparam.pdf

http://www3.nccu.edu.tw/~tsaich/CA/SPSS_5.pdf

http://archived.chns.org/s.php@page=11&id=34&id2=1222.html

http://aiwwu66.blog.163.com/blog/static/13628627920111254739501/

[Permutation test ref]

http://www.hanlongfei.com/statistical%20computing%20with%20r/2014/12/23/permutationtest/

https://www.plob.org/article/3176.html

[Run test]

https://onlinecourses.science.psu.edu/stat414/node/329

http://web.nchu.edu.tw/~finmyc/stat17p.pdf

http://www.math.nsysu.edu.tw/~lomn/homepage/class/92/The%20Runs%20Test/The%20Runs%20Test.pdf

https://www.youtube.com/watch?v=YWlod6Jdu-k

https://www.youtube.com/watch?v=Ps7RwTgB4HA

http://www.r-web.com.tw/stat/step1.php?method=one_sample_runs_test

http://www3.nccu.edu.tw/~soci1005/Ch11.pdf

[常態性檢定]

https://www.shiyanlou.com/courses/reports/1296530

http://westerly-lzh.github.io/cn/2014/02/Statistic-Test/