0330作業

統學

2021/3/29

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

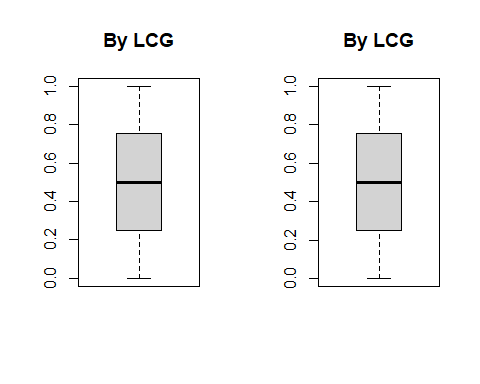
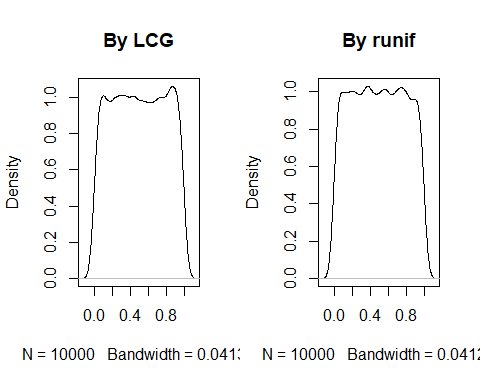
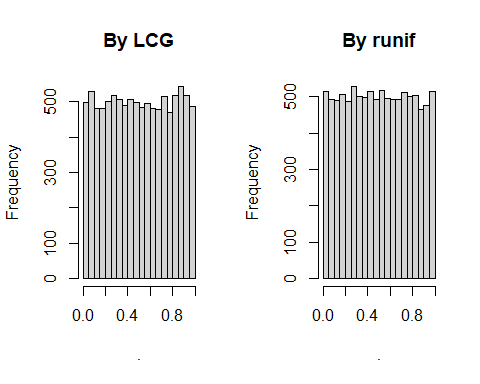
## √ ggplot2 3.3.2 √ purrr 0.3.4  
## √ tibble 3.0.4 √ dplyr 1.0.2  
## √ tidyr 1.1.2 √ stringr 1.4.0  
## √ readr 1.3.1 √ forcats 0.5.0

## Warning: package 'tibble' was built under R version 4.0.3

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

一開始利用製造自抽樣的樣本。  
圖左邊為以LCG製造的樣本所畫的直方圖及以該樣本所估計的Density Plot，與右圖相比起來十分相似，從盒狀圖來看，也可看出兩組資料十分相近。

# 此為LCG生成所用之演算法  
lcg <- function(a,c,m,run.length,seed) {  
 x <- rep(0,run.length)  
 x[1] <- seed  
 for (i in 1:(run.length-1)) {  
 x[i+1] <- (a\*x[i] + c) %% m  
 }  
 U <- x/m   
 return(list(x=x,U=U))  
}

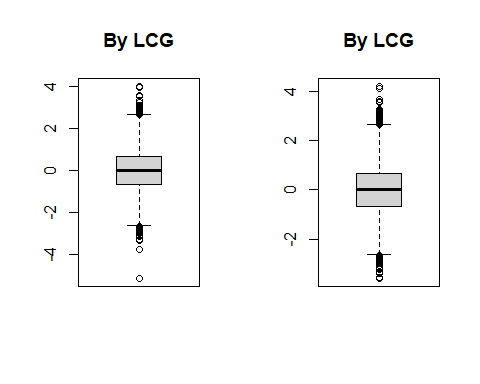
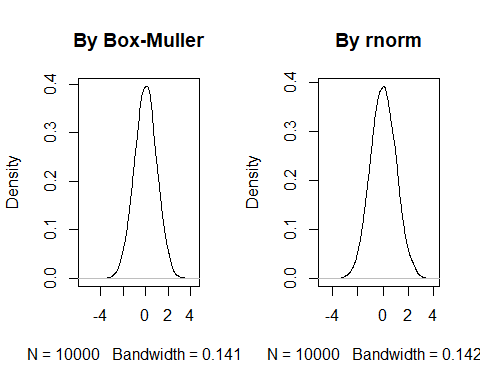
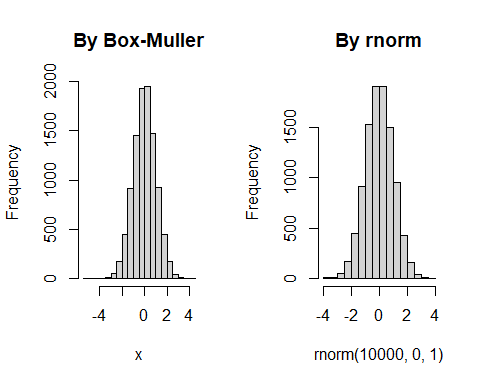


# Problem 1

## (a)

以法製造出10000個標準常態  
圖左邊為以LCG製造的方法所畫的直方圖及DensityPlot，與右圖相比起來十分相似。  
從盒狀圖來看，也可看出兩組資料十分相近，僅在Q1,Q3以外，法較些微多一些。

# 此為Box-Muller所用之演算法  
fnc = function(U1,U2){  
 sqrt(-2\*log(U1))\*cos(2\*pi\*U2)  
}



## (b)

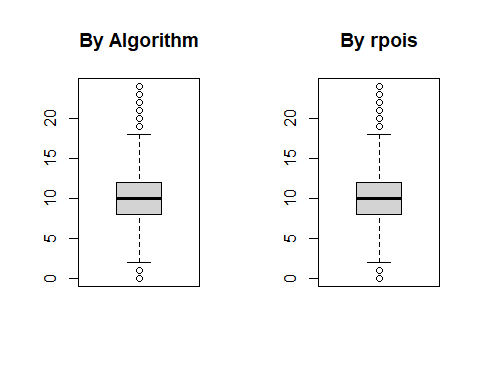
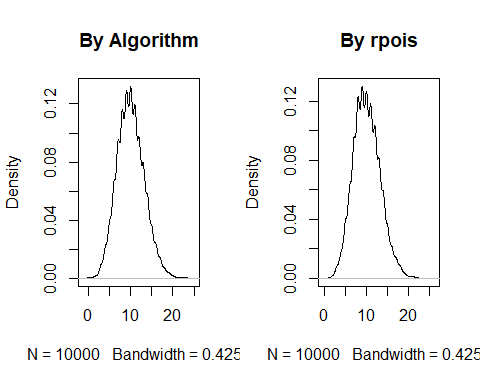
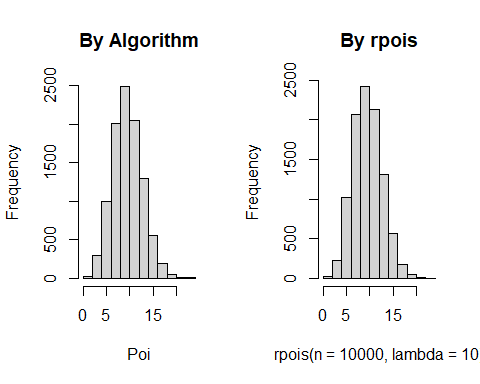
需要用到指數分配，放在第二題結束的地方

# Problem 2

## (a)

圖左邊為自製以上課所教的演算法所得到的樣本，所畫的直方圖及DensityPlot，與右圖相比起來十分相似。  
從盒狀圖來看，也可看出兩組資料十分相近。

# 此生成Poisson分配所用之演算法  
  
Poisson = function(mu,n){  
 X = sapply(1:n, function(a){  
 t = 0 ; X = 0 ; lambda = mu  
 while (t<1) {  
 U = sample(Uni$U,1)  
 t = t - (1/lambda)\*log(U)  
 X = X+1  
 }  
 X = X-1  
 X  
 })   
   
 return(X)  
}

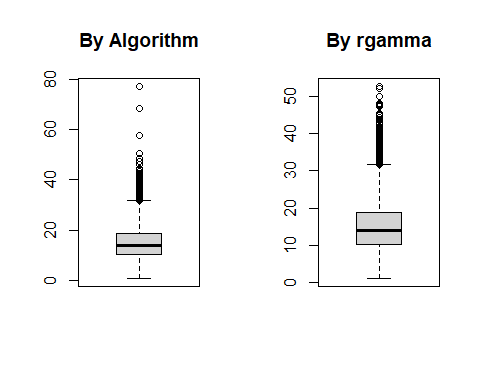
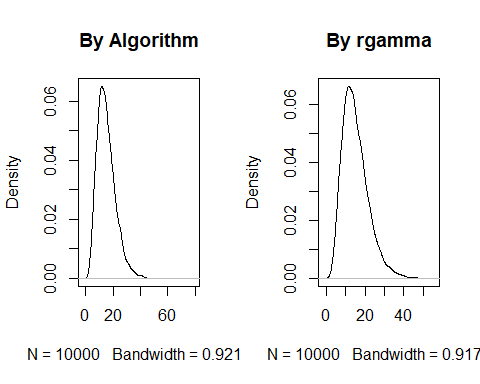
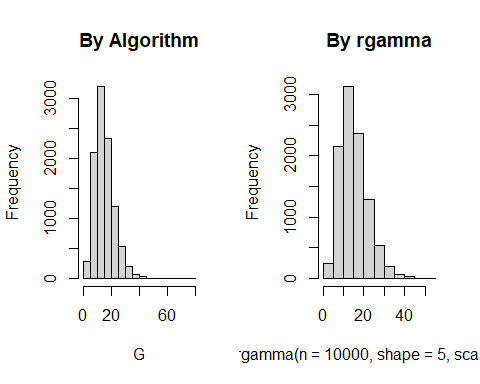


## (b)

先以上課所教的演算法，生成，再將連加，即可得到的樣本。  
圖左所生成樣本畫的直方圖及DensityPlot，與右圖相比起來十分相似。  
從盒狀圖來看，也可看出兩組資料十分相近，但自己生成的Gamma有較易有偏離期望值很多的值產生，也因此自生成樣本的全距較大。

# 此為生成Exponential分配所用之演算法  
EXP = function(mu,n){  
 lambda =mu  
 N = sapply(1:n, function(a){  
 U = sample(Uni$U,1)  
 X = -(1/lambda)\*log(U)  
 })  
 N  
}

# 此為生成Gamma 分配所用之演算法  
GAM = function(a,b,n){  
 i = 1  
 x = matrix(0,nrow = n)  
 repeat{  
 if(i>a) break  
 y = EXP(b,n) %>% as.matrix()  
 x = x+y   
 i = i+1  
 }  
 return(x)  
}



## Problem 1

### (b)

以下為的演算法

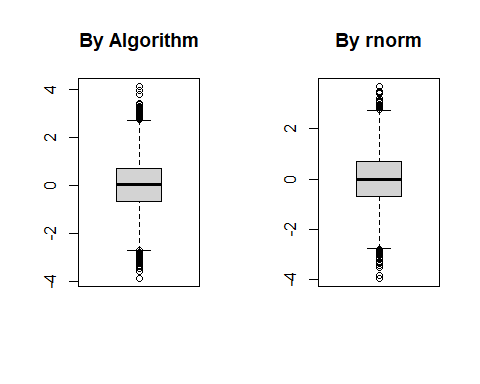
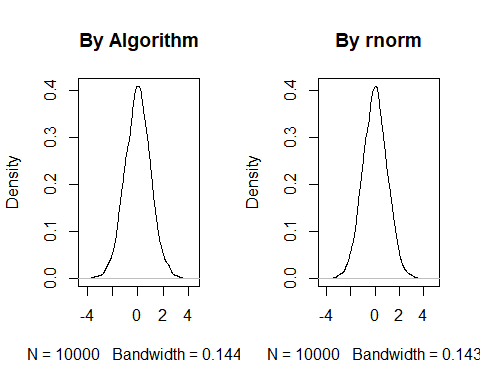
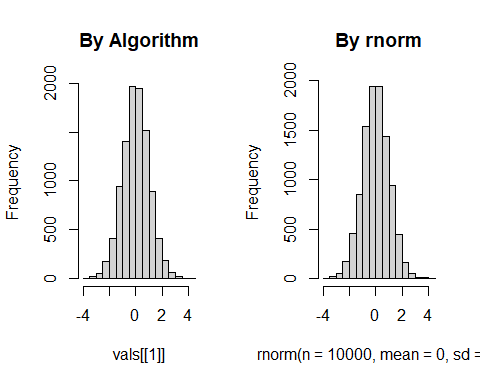
E = EXP(1,10000)  
  
NormalAR = function(c,n){   
 f = function(z){ #設定f為標準常態分配   
 sqrt(2/pi)\*exp(-z^2/2)   
 }  
 g = function(z){ #設定g為exp(1)   
 exp(-z)  
 }  
   
 f\_z = as.vector(rep(0,n))  
 i = 0 #設定i迭代至所需樣本數  
 b = 1   
 a = c() #設定a,b紀錄該次是否接受，計算Acceptance Rate  
 while (i< n){  
 b = b+1  
 z = -log(sample(Uni$U,1))   
 u1 = sample(Uni$U,1)   
 a[b] = ifelse(u1 < f(z)/(c\*g(z)),"Y","N")  
   
 if(u1 < f(z)/(c\*g(z))){  
   
 u2 = sample(Uni$U,1)  
 if(u2 < 0.5) { Z = z } else {Z = -z }  
 i = i+1  
 f\_z[i] = Z  
 }  
 }  
 return(list(f\_z,a))  
}

## Acceptance Rate

首先設定，可得到此時約為75%。  
但若是改變c，令，則此時的下降至約38%。

## [1] "Acceptance\_Rate\_1 : 74.86%"

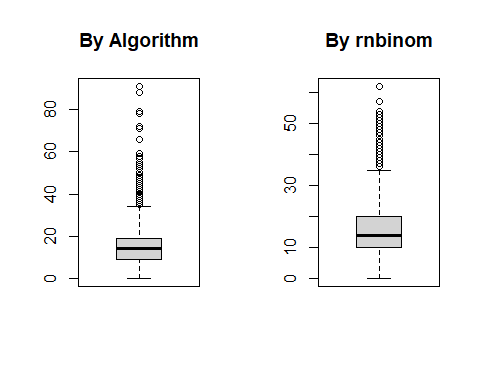
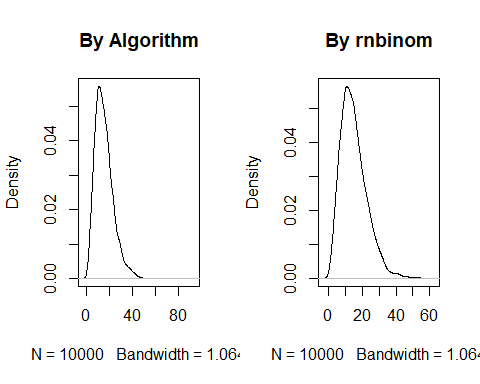
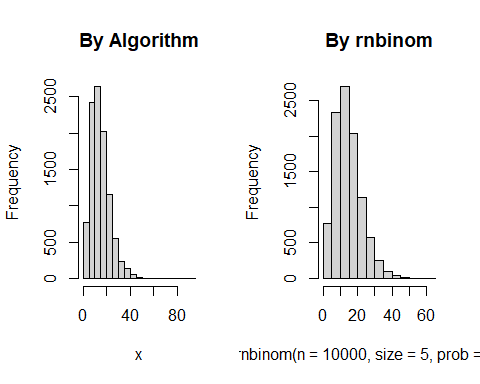
## [1] "Acceptance\_Rate\_2 : 37.83%"

先生成，圖左所生成樣本畫的直方圖及DensityPlot，與右圖相比起來十分相似。  
從盒狀圖來看，也可看出兩組資料十分相近。 

# Problem 3

## (c)

# 樣本產生方式，將Gamma生成樣本帶入Poisson分配  
  
G = GAM(a = 5,b = 1/3,n = 10000)  
x = matrix(0,nrow=10000)  
for (i in 1:10000) {  
 x[i,] = Poisson(mu = G[i],n=1)  
}

由上兩題可得知此混和分配為負二項分配，將生成樣本與比較。  
圖左所生成樣本畫的直方圖及DensityPlot，與右圖相比起來十分相似。  
從盒狀圖來看，也可看出兩組資料十分相近，但自生成的樣本有偏離較大的值出現。 

可發現自生成樣本的平均數約為15，變異數約為60，  
與使用所產生的樣本結果相似。

## [1] "Mean from the samples by myself: 15.0759"

## [1] "Mean from the samples by rnbinom(): 15.0826"

## [1] "Variance from the samples by myself: 64.649404130413"

## [1] "Variance from the samples by rnbinom(): 59.2290229022902"

# Problem 4

## (b)

由(a)小題可得知其機率密度函數，其生成樣本之演算法解釋如下： 1. 自抽一項, 2. 若，則生成，若則生成

vals <- NormalAR(c = sqrt(2/pi)\*exp(1/2),n = 10000)  
  
N = 10000   
U =runif(N)  
rand.samples = rep(NA,N)  
for(i in 1:N){  
 if(U[i]<0.75){  
 rand.samples[i] = sample(vals[[1]],1)  
 }else{  
 rand.samples[i] = sample(vals[[1]],1)+3 #自N(0,1)平移3，變成N(3,1)  
 }  
}

生成樣本後，畫出該分配的直方圖及DensityPlot，可發現在在0,3的地方有雙峰的現象，符合我們一開始的想像。 