# hw 4 solution

Autumn 2022

**문제 1.**

1. 문제에서 지정한 선형합동법 알고리즘 함수를 정의한다. 난수가 0~1사이에 위치하도록 나머지에 $2^{32}$를 나누어주었다.

```
superduper = function(n){
  x = 0
  for (i in 1:n+1){
    x[i] = (69069 * x[i-1] + 1) %% (2^32)
  }
  return (x[-1]/2^32)
}
```

2. RANDU로부터 난수 100개를 생성하였다.

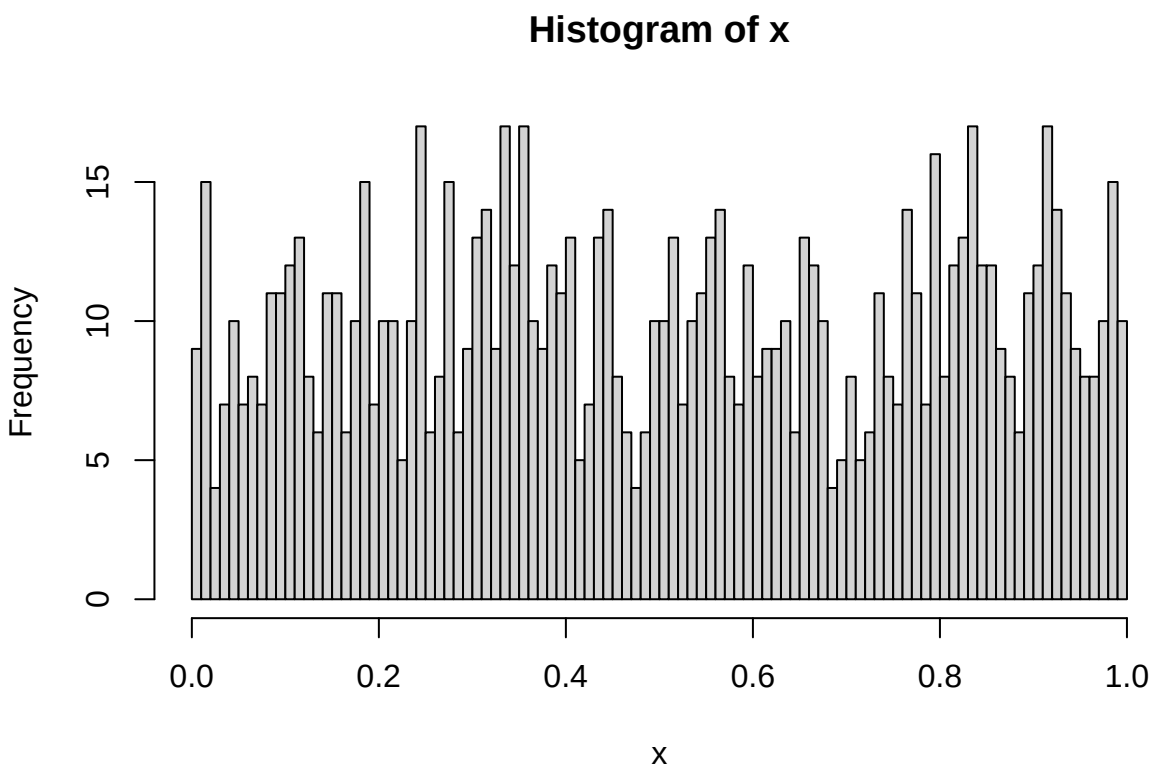
```
N = 1000
x = superduper(N)
head(x)
```

```
## [1] 2.328306e-10 1.608161e-05 1.107409e-01 7.630801e-01 1.799780e-01
## [6] 9.028781e-01
```

3. 생성한 난수들이 $U(0,1)$에서의 확률 난수인지 확인하기 위해 분포를 그려보고 적합도검정, 독립성 검정을 수행해 보았다.

(1) 생성된 난수들은 0~1 사이에 비교적 고르게 분포해 보인다.

```
hist(x, breaks = 100)  # 산점도
```




(2) 카이제곱 적합도 검정은 가설 ($H_0$: 생성된 난수들은 $[0,1]$사이의 균일분포를 따른다. vs $H_1$: not $H_0$)에 따라 검정하였다. 카이제곱 적합도 검정의 결과는 $[0,1]$의 구간을 어떻게 나누느냐에 따라 p-value의 값이 약간씩 달라지는데, 여기서는 적당히 10개의 구간으로 나누었고 p-value가 0.05보다 크므로 $H_0$를 기각하지 않았다 (난수들이 균일분포를 따름).

```
k = 11

range = seq(0, 1, length = k)

# [0, 1]을 k등분
n = table(cut(x, range))
n  # 구간별 난수의 갯수
```

```
##
##   (0,0.1] (0.1,0.2] (0.2,0.3] (0.3,0.4] (0.4,0.5] (0.5,0.6] (0.6,0.7] (0.7,0.8]
##        89        99        96       124        86       105        86        93
## (0.8,0.9]   (0.9,1]
##       108       114
```

```
# 구간별로 난수의 갯수 세기
W = ((k - 1) / N) * sum( (n - ( N/(k - 1) ) )^2)
pchisq(W, df = k-2, lower.tail = FALSE)  # p-value
```

```
## [1] 0.108791
```

(3) 또 다른 적합도 검정으로 콜모고로프-스미르노프 적합도 검정을 할 수 있는데(`ks.test` 함수 사용), 그 결과 생성한 난수들이 균일분포를 따르지 않는다고 할 수 있다.

```
u = runif(N)
ks.test(x, u)
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  x and u
## D = 0.066, p-value = 0.02566
## alternative hypothesis: two-sided
```

(4) 마지막으로 독립성 검정인 런 검정 (run test)를 가설 ($H_0$: 난수들이 서로 독립이다. vs $H_1$: not $H_0$) 하에서 시행하였고, p-value가 0.899로 생성한 난수들이 서로 독립이라고 할 수 있다.

```
library(snpar)
runs.test(x)
```

```
##
##  Approximate runs rest
##
## data:  x
## Runs = 499, p-value = 0.8993
## alternative hypothesis: two.sided
```

**파이썬**

파이썬은 카이제곱 검정까지 수행하였고, 검정 결과로 생성한 난수가 균일분포를 따른다고 할 수 있다.

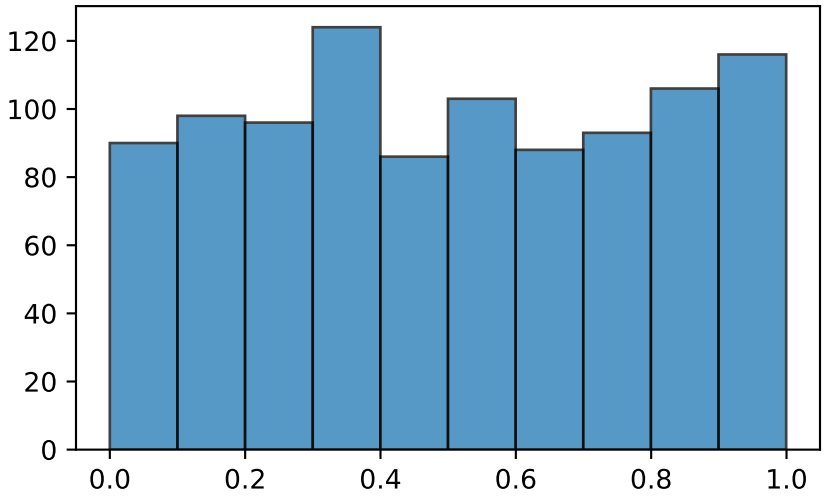
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

def superduper_py(n):  # 난수발생 함수 정의
  x = np.array([0])
  for i in range(1, n):
    x = np.append(x, (69069 * x[i-1] + 1) % pow(2,32))

  np.delete(x, 0)
  return x/pow(2,32)

N = 1000
x = superduper_py(N)
x[0:10]
```

```
## array([0.00000000e+00, 2.32830644e-10, 1.60816126e-05, 1.10740898e-01,
##        7.63080108e-01, 1.79978035e-01, 9.02878134e-01, 8.89840406e-01,
##        3.87011191e-01, 4.75942640e-01])
```

```
fig = plt.figure(figsize=(5,3))  # 난수의 분포 확인
fig.set_facecolor('white')
plt.hist(x, edgecolor='k', alpha = 0.75)
```

```
## (array([ 90.,  98.,  96., 124.,  86., 103.,  88.,  93., 106., 116.]), array([0.        , 0.09995122,
##        0.49975609, 0.59970731, 0.69965853, 0.79960975, 0.89956097,
##        0.99951219]), <a list of 10 Patch objects>)
```

```
plt.show()
```




```
bins = np.arange(0, 1, 0.1)      # 도수분포구간 정하기
hist, bins = np.histogram(x, bins)
print(hist)   # 구간
```

```
## [ 90  98  96 124  86 105  86  93 108]
```

```
print(bins)   # 구간에 해당하는 도수
```

```
## [0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9]
```

```
k = 10
W = (k/N) * sum(pow(hist - (N/k), 2)) # 카이제곱 검정통계량
1-stats.chi2.cdf(W, k-1)              # p-value
```

```
## 0.19904546447938287
```

**문제 2.**

1. 확률 모의실험을 위해 먼저 각 10,000개의 균일난수를 아래와 같이 생성하였다.

```
set.seed(123)
u1 = runif(10000)
u2 = runif(10000)
```

2. (a) 독립인 두 확률변수의 분산의 합과 합의 분산은 이론적으로 같으며 $(2 \times \frac{1}{12})$, 경험적 추정치 역시 참값과 비슷한 값을 출력하였다.

```
# (a)
2 * (1 / 12)  # 참값
```

```
## [1] 0.1666667
```

```
var(u1 + u2)
```

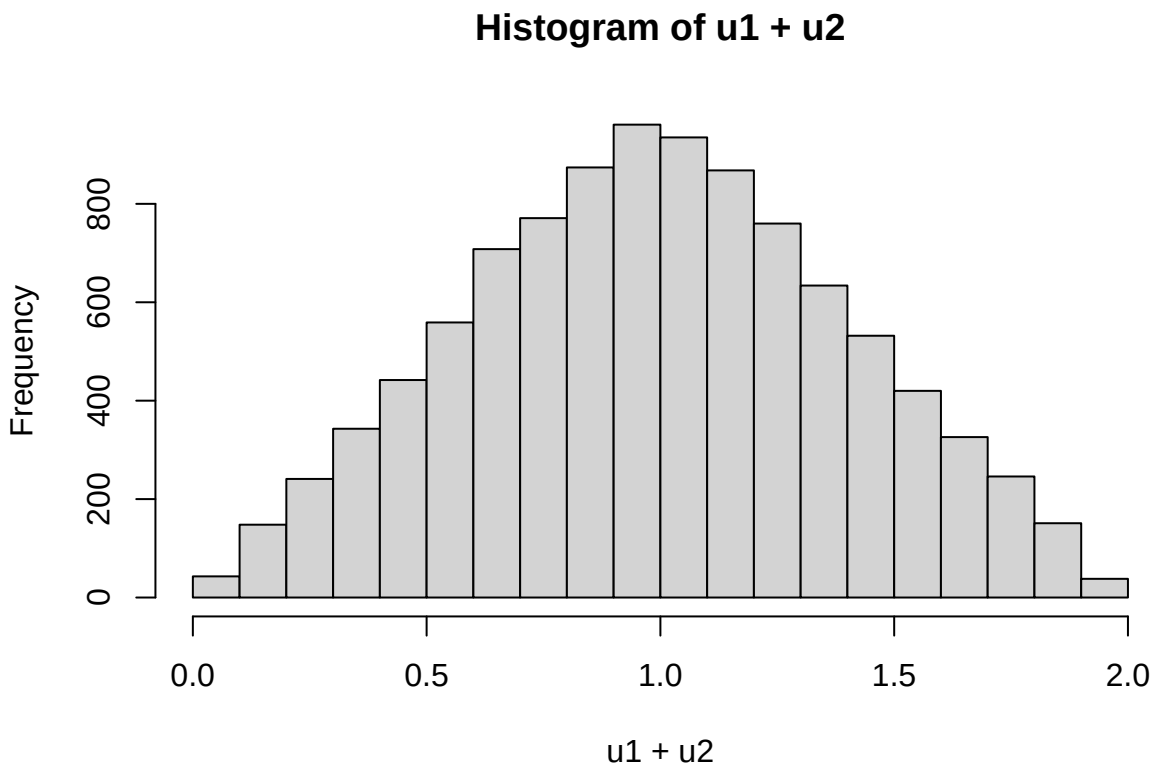
```
## [1] 0.1623921
```

```
var(u1) + var(u2)
```

```
## [1] 0.1653023
```

(b) $X = U_1 + U_2$라 하면 $X$의 분포는 $f(x) = xI(0 \le x \le 1) + (2-x)I(1 \le x \le 2)$를 따른다. 따라서 참값은 $P(X \le 1.2) = \int_0^{1.2} f(x)dx = 0.68$ 이며, 경험적 추정치는 100개 발생한 난수에 대해 $\frac{1}{10000}\sum_{i=1}^{10000} I(u_1 + u_2 \le 1.2)$로 계산할 수 있다. 결과에 따라 참값과 경험적 추정치가 비슷하게 나타남을 볼 수 있다.

```
# (b)
hist(u1+u2)  # distribution of U1 + U2
```




```
f = function(x){ x * (0 <= x) * (x <= 1) + (2 - x) * (1 <= x) * (x <= 2) }

integrate(f, 0, 1.2)     # true value
```

```
## 0.68 with absolute error < 1.8e-05
```

```
mean( (u1 + u2) <= 1.2 ) # empirical estimator
```

```
## [1] 0.6893
```

**파이썬**

파이썬에서도 균일난수 u1와 u2를 발생하고 u1+u2의 분산, u1의 분산과 u2의 분산을 비교하였고 그 결과가 비슷함을 보였다. 그리고 $P(X \le 1.2)$의 경험적 추정치 역시 약 0.68으로 참값과 비슷함을 보였다.

```
import random
import scipy.integrate as sci

n = 10000
u1 = []
u2 = []
```

```
for i in range(0, n):
  u1.append(random.uniform(0, 1))
  u2.append(random.uniform(0, 1))

u1 = np.array(u1)
u2 = np.array(u2)

# (a)
np.var(u1 + u2)
```

```
## 0.16570408215946297
```

```
np.var(u1) + np.var(u2)

# (b)
```

```
## 0.1664661568829623
```

```
fx = lambda x: x * (0<=x) * (x<=1) + (2-x) * (1<=x) * (x<=2)

sci.quad(fx, 0, 1.2)
```

```
## (0.68, 7.549516567451065e-16)
```

**문제 3.**

두 개의 주사위를 던질 때 24번 이내에 두 주사위가 모두 6이 나올 확률의 추정값은 약 0.493으로 참값이 0.49와 비슷하다.

```
set.seed(1000)
dice1 = dice2 = 1:6
iter = 100000

out = c()
for(i in 1:iter){
  dice_result = sapply(1:24, function(j){
                sam1 = sample(dice1, 1); sam2 = sample(dice2, 1)
                return(c(sam1, sam2))
                } )

  out[i] = sum(colSums(dice_result) == 12) >= 1  # 두 주사위 모두 6이 나왔는지 확인
}
sum(out) / iter
```

```
## [1] 0.49291
```

**파이썬**

파이썬으로 구현한 결과도 약 0.49로 나타난다.

```
dice1 = np.arange(1,7,1)
dice2 = np.arange(1,7,1)
maxiter = 100000
out = []

for i in range(0, maxiter):
  dice_result = np.array([[0, 0]])
  for j in range(0, 24):
```

```
    sam1 = random.choice(dice1)
    sam2 = random.choice(dice2)
    dice_result = np.append(dice_result, np.array([[sam1, sam2]]), axis = 0)
  out.append( sum(dice_result.sum(axis=1) == 12) >= 1 )

sum(out) / maxiter
```

```
## 0.49045
```