Statistical Computing and Simulation: HW1

Deparment of Statistics, NCCU
葉佐晨
112354016@nccu.edu.tw

高崇哲
112354@nccu.edu.tw

2024-03-07

Statistical Computing and Simulation

Assignment 1, Due March 15/2024

1. (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.)

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

$$u_i = \frac{x_i}{30269} + \frac{y_i}{30307} + \frac{z_i}{30323} \pmod 1$$

   with $x_i = 171x_{i-1} \pmod{30269}$, $y_i = 172y_{i-1} \pmod{30307}$, $z_i = 170z_{i-1} \pmod{30323}$. Compare the results in (a) and (b), and discuss your findings.

```r
library(magrittr)
library(extraDistr)
```

```r
midSquare <- function(size = 1){
  initSample <- function(){sample(0:999999, size = 1)}

  num <- initSample()
```

```r
  nums <- c(num)

  for (i in seq(size)){
    num <- num^2 %>%
      format(scientific = FALSE) %>%
      substr(start = 4, stop = 9) %>%
      ifelse(. == "", initSample(), .) %>%
      as.integer()

    num <- ifelse(num == nums[length(nums)], initSample(), num)
    nums <- c(nums, num)
  }

  return(nums)
}
```

```r
randSeq <- midSquare(size = 10000)
ks.test(randSeq, "pdunif", 0, 999999)
```

```
## Warning in ks.test.default(randSeq, "pdunif", 0, 999999): ties should not be
## present for the Kolmogorov-Smirnov test
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  randSeq
## D = 0.13611, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```r
LCG <- function(size, mod, mul, seed = NULL){
  nums <- c()
  if (!is.null(seed)){set.seed(seed)}

  xyz <- sapply(mod, function(x){sample(0:x, size = 1)})

  for (i in seq(size)){
    u <- sum(xyz / mod)
    nums <- c(nums, u)
    xyz <- (xyz * mul) %% mod
```

```r
  }

  return(nums)
}
```

```r
mod <- c(30269, 30307, 30323)
mul <- c(172, 171, 170)
randomSeq <- LCG(size = 10000, mod = mod, mul = mul)

ks.test(randomSeq, "punif", 0, 3)
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  randomSeq
## D = 0.18528, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

3. 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 $\alpha = 0.05$ 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.

```r
library(nortest)
```

```r
norTest <- function(testList, n, seed = NULL){
  if (!is.null(seed)) {set.seed(seed)}

  res <- rbind(
    sapply(testList, function(test){test(rnorm(n))[["p.value"]]}),
    sapply(testList, function(test){test(rt(n, df = 10))[["p.value"]]}),
    sapply(testList, function(test){test(rt(n, df = 20))[["p.value"]]})
  )

  colnames(res) <- c(c("ad", "cvm", "lillie", "pearson", "sf"))
  rownames(res) <- c("N(0,1)", "t(df=10)", "t(df=20)")
```

```r
  return(res)
}

countReject <- function(n_trial, n, alpha){
  replicate(n = n_trial,
            norTest(testList, n = n) <= alpha) %>%
    apply(MARGIN = c(1, 2), sum)
}
```

```r
n <- c(10, 50, 100)
testList <- c(ad.test, cvm.test, lillie.test, pearson.test, sf.test)
n_trial <- 1000
alpha <- 0.05
```

```r
rejList <- lapply(n, function(n){countReject(n_trial = 1000, n = n, alpha = 0.05)})
names(rejList) <- paste0("n=", n)
print(rejList)
```

```
## $`n=10`
##          ad cvm lillie pearson sf
## N(0,1)   57  48     48      71 63
## t(df=10) 75  61     72      71 82
## t(df=20) 58  56     60      72 79
##
## $`n=50`
##           ad cvm lillie pearson  sf
## N(0,1)    47  47     45      56  68
## t(df=10) 126 100     71      62 195
## t(df=20)  75  82     68      65 105
##
## $`n=100`
##           ad cvm lillie pearson  sf
## N(0,1)    54  51     54      59  56
## t(df=10) 187 165    110      63 289
## t(df=20)  89  74     83      60 147
```

5. (a) Use the search engine to download the first one million digits of pi (for example, https://www.piday.org/million/) and check via graphic tools if the numbers violate the assumption of random numbers.

(b) Apply the appropriate tools to test if the random numbers from

(a) satisfy the assumption of random numbers.

```r
library(randtoolbox)
```

```
## Loading required package: rngWELL

## This is randtoolbox. For an overview, type 'help("randtoolbox")'.
```

```r
pi1M <- readLines("Pi1MDP.txt", warn = FALSE) %>%
  strsplit(split = "") %>%
  unlist() %>%
  as.integer()
```

```r
isRandom <- function(seq, alpha = 0.05){
  tb <- table(pi1M)
  testChi <- tb %>% chisq.test
  print(testChi)
  isUnif <- ifelse(testChi[["p.value"]] < alpha, FALSE, TRUE)

  cat(rep("*", 60), sep = "")

  r <- diff(range(seq))
  isIndept <- ifelse(gap.test(seq/r, echo = TRUE)[["p.value"]] < alpha,
                     FALSE, TRUE)

  return(isUnif & isIndept)
}
```

```r
isRandom(pi1M, alpha = 0.05)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  .
## X-squared = 5.5091, df = 9, p-value = 0.7879
##
## ************************************************************
##           Gap test
```

```
##
## chisq stat = 44, df = 20, p-value = 0.0013
##
##       (sample size : 1000000)
##
## length   observed freq       theoretical freq
## 1             125596              125000
## 2             62325           62500
## 3             31348           31250
## 4             15788           15625
## 5             7900            7812
## 6             3857            3906
## 7             1888            1953
## 8             925             977
## 9             444             488
## 10            232             244
## 11            148             122
## 12            60              61
## 13            20              31
## 14            18              15
## 15            6           7.6
## 16            9           3.8
## 17            3           1.9
## 18            0           0.95
## 19            0           0.48
## 20            1           0.24
## 21            1           0.12

## [1] FALSE
```