STAT 812: Computational Statistics

Random Number Generator and Monte Carlo

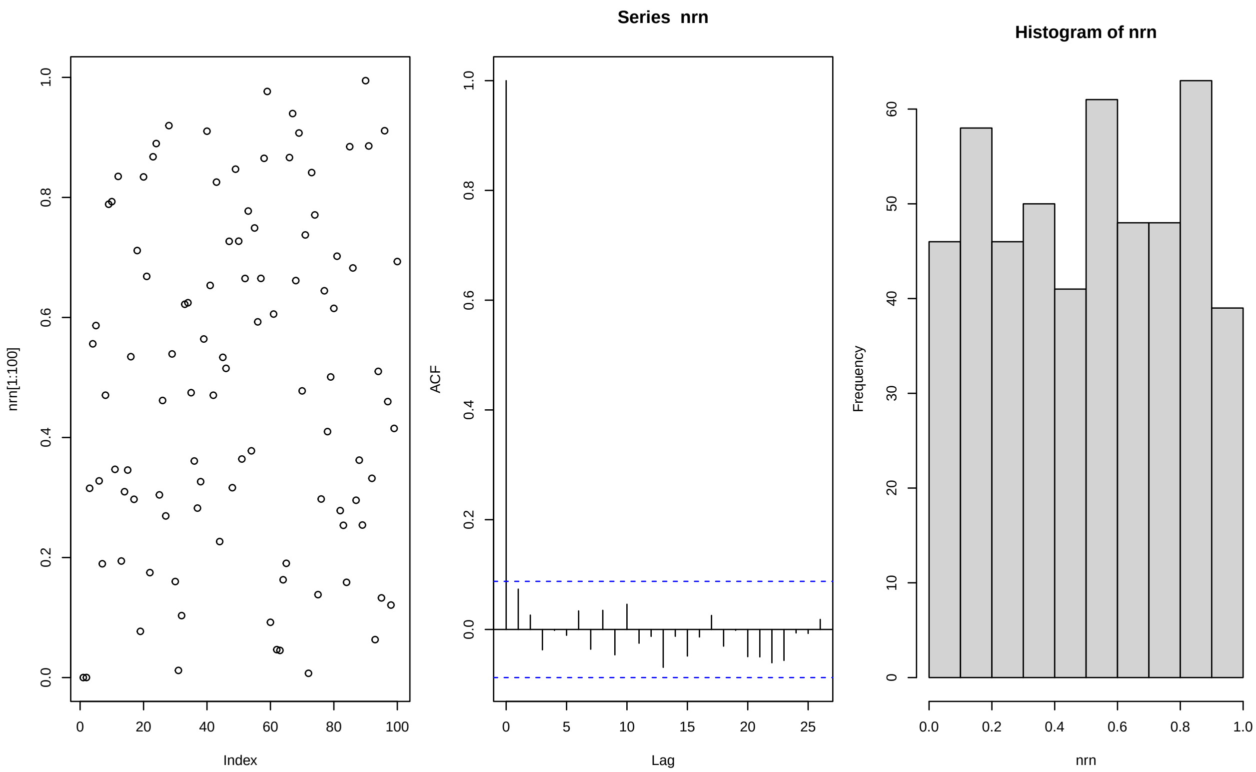
Longhai Li

2024-09-16

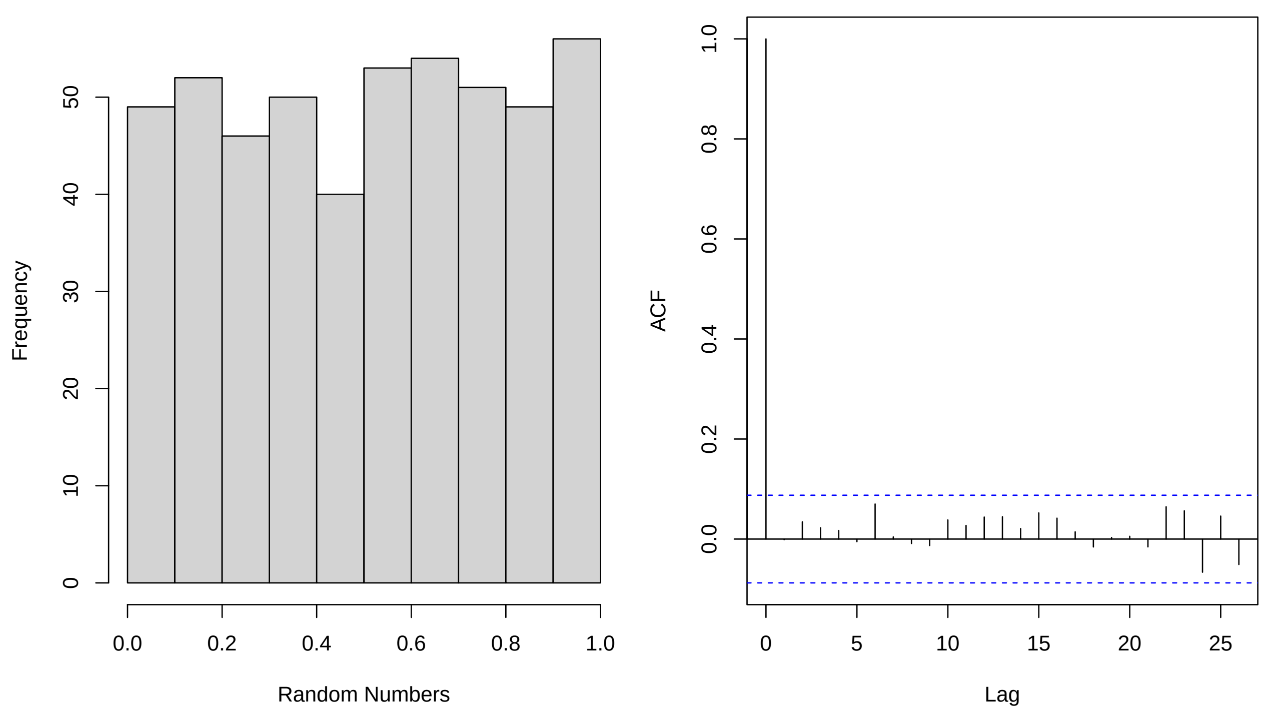
Contents

1 Pseudo random numbers

```
A <- 7^5
M <-  2^31-1

N <- 500
rn <- rep (0, N)
rn[1] <- 10
for (i in 2:length (rn))
{
    rn[i] <- (A * rn[i-1] ) %% M
}

nrn <- rn/(M-1)
par(mfrow=c(1,3),mar=c(4,4,3,1))
plot (nrn[1:100])
acf (nrn)
hist (nrn)
```

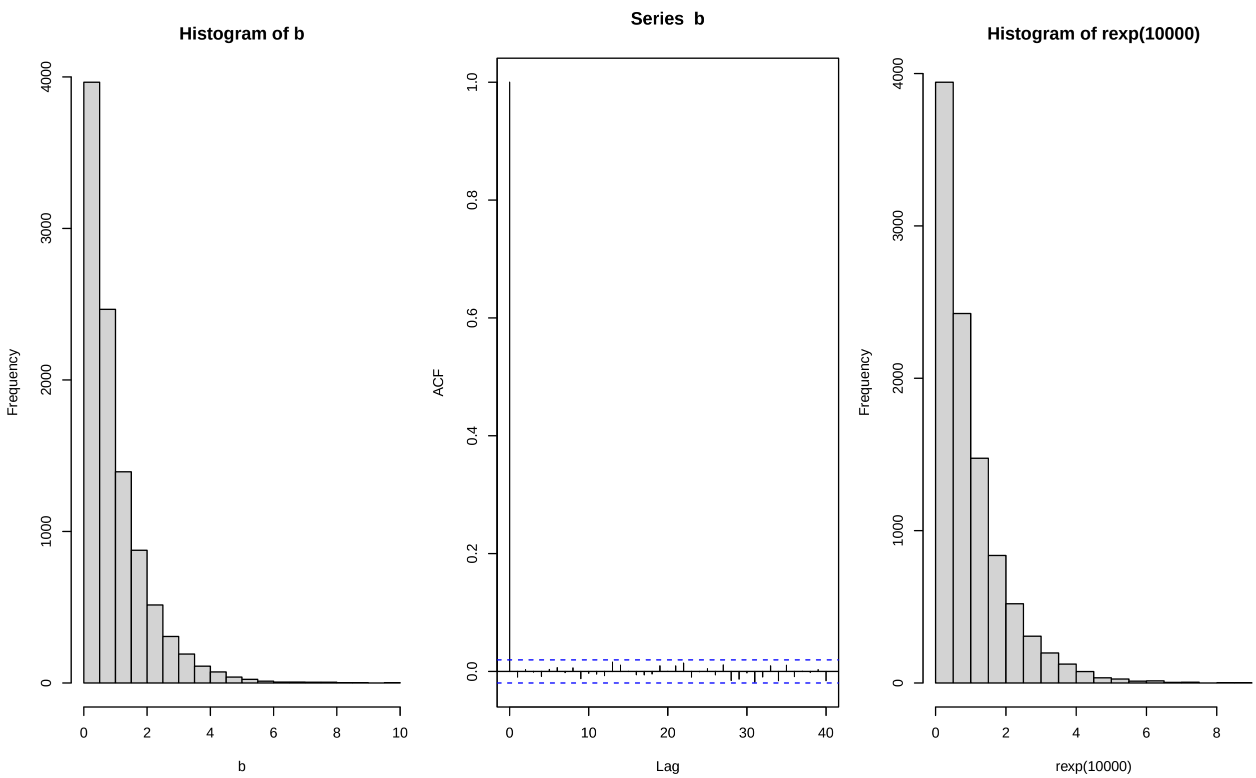



```r
n <- 500
a <- runif(n)
par(mfrow=c(1,2),mar=c(4,4,3,1))
hist(a,xlab="Random Numbers",main="")

acf(a,main="")
```



2 Inverting CDF

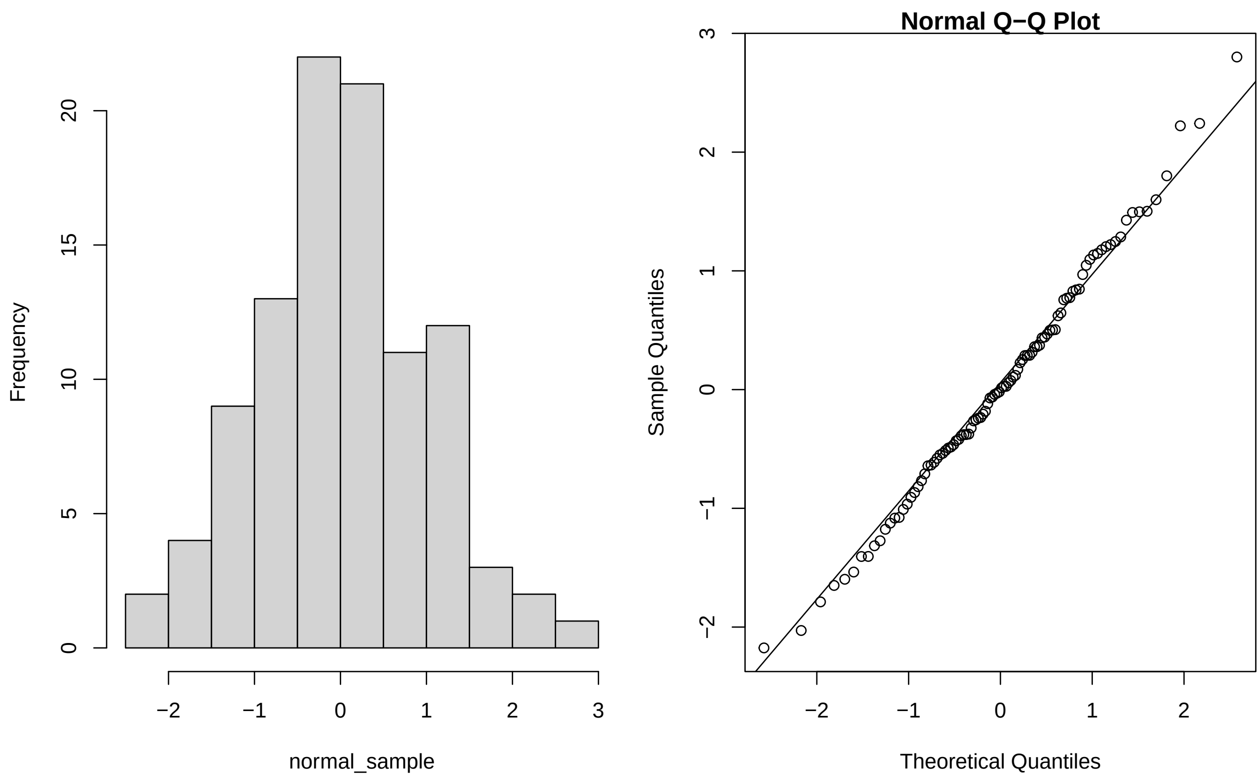
```
# generate exponenail random numbers

#use method of inverse cdf to generate iid sample from exp(1)
gen_exp <- function(n)
{
    #generate unif(0,) random numbers
    u <- runif(n)
    #transform the random numbers
    -log(1-u)
}

b <- gen_exp (10000)
par(mfrow=c(1,3),mar=c(4,4,3,1))
hist (b)
acf (b)

## r built in generators
hist (rexp (10000))
```



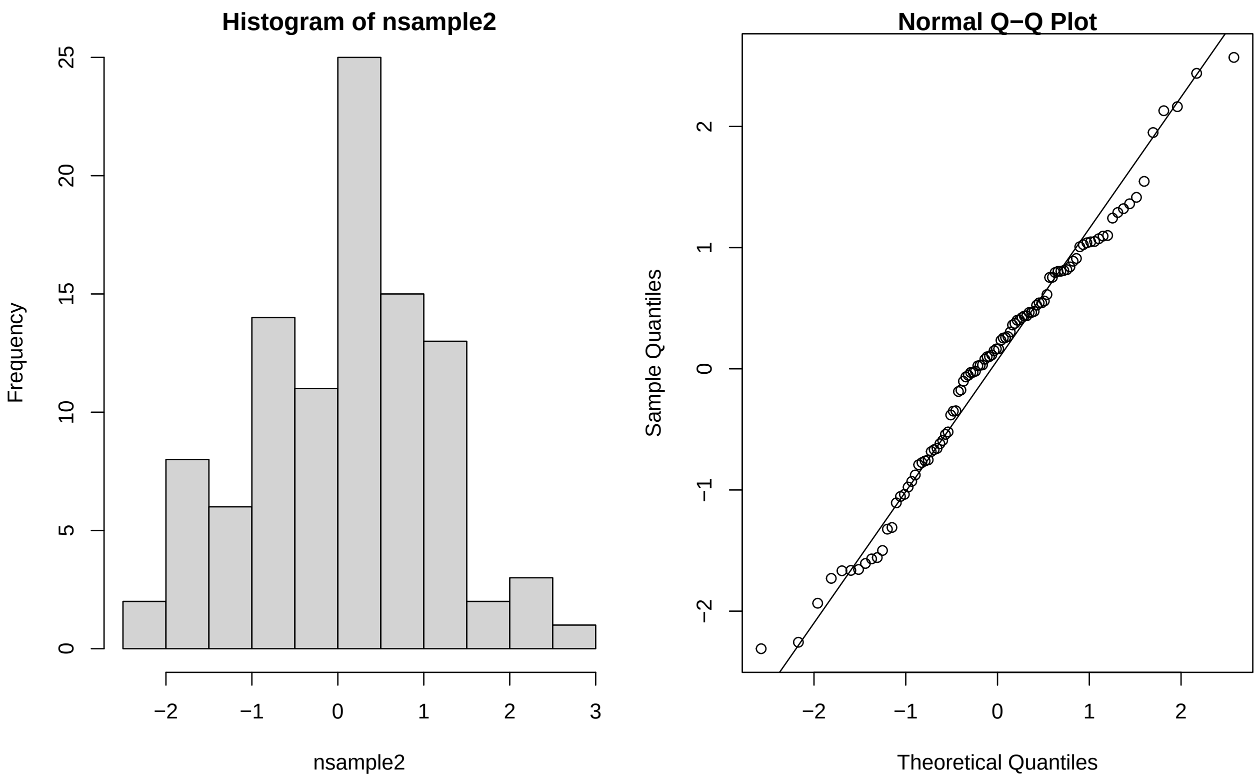
3 A Special Transformation for Generating Normal Sample

```
gen_normal <- function(n)
{
    #calculates size of random samples, which is greater than half of n
    size_sample <- ceiling(n/2)
```

```
    R <- sqrt(2*rexp(size_sample))
    theta <- runif(size_sample,0,2*pi)

    X <- R*cos(theta)
    Y <- R*sin(theta)

    c(X,Y)[1:n]
}

normal_sample <- gen_normal(100)

par(mfrow=c(1,2),mar=c(4,4,1,1))

hist(normal_sample,main="")

qqnorm(normal_sample)
qqline(normal_sample)
```




```
nsample2 <- rnorm (100)
hist (nsample2)
qqnorm (nsample2)
qqline(nsample2)
```



4 Demonstration of CLT and LLN

5 An Example of Monte Carlo for Estimating π

```
#### an application of monte carlo method in estimating pi

# n is the number of samples drawn uniformly from the rectangle (-1,1)  * (-1,1)
# an estimate of pi is returned
pi_est_mc <- function(n)
{
    #X and Y are independent, each with marginal distribution unif(-1,1)
    X <- runif(n,-1,1)
    Y <- runif(n,-1,1)

    Z <- 4 * (X^2 + Y^2 <= 1)
    mu <- mean (Z)
    error <- 1.96 * sd (Z) /sqrt (n)
    list (pi.est = mu, error.95perc = error, ci.95perc = mu + c(-error, error))
}

pi_est_mc (100)
```

```
## $pi.est
## [1] 3.4
##
## $error.95perc
## [1] 0.2813543
##
```

```
## $ci.95perc
## [1] 3.118646 3.681354
```

```r
pi_est_mc (10000)
```

```
## $pi.est
## [1] 3.112
##
## $error.95perc
## [1] 0.03258398
##
## $ci.95perc
## [1] 3.079416 3.144584
```

```r
pi_est_mc (100000)
```

```
## $pi.est
## [1] 3.14024
##
## $error.95perc
## [1] 0.01018423
##
## $ci.95perc
## [1] 3.130056 3.150424
```

```r
pi_est_mc (10000000)
```

```
## $pi.est
## [1] 3.140321
##
## $error.95perc
## [1] 0.001018383
##
## $ci.95perc
## [1] 3.139302 3.141339
```