Statistical Computing Homework 2, Chapter 1

Ziqi Yang

14 September, 2018

Contents

Exercise 2: Monte Carlo methods

R Code

```
N <- c(100 ,1000, 10000);  Ts <- c(0.0, 0.67, 0.84, 1.28, 1.65, 2.32, 2.58, 3.09, 3.72)
experiment <- array( 0, dim = c(length(N), length(Ts), 100) )

for (expe in 1:100) {
  for (t in 1:length(Ts)) {
    for (n in 1:length(N)) {
      temp <- rnorm(N[n], mean = 0, sd = 1)
      experiment[n ,t , expe] <- sum( ifelse( temp <= Ts[t], 1, 0 ) ) / N[n]
    }
  }
}
```

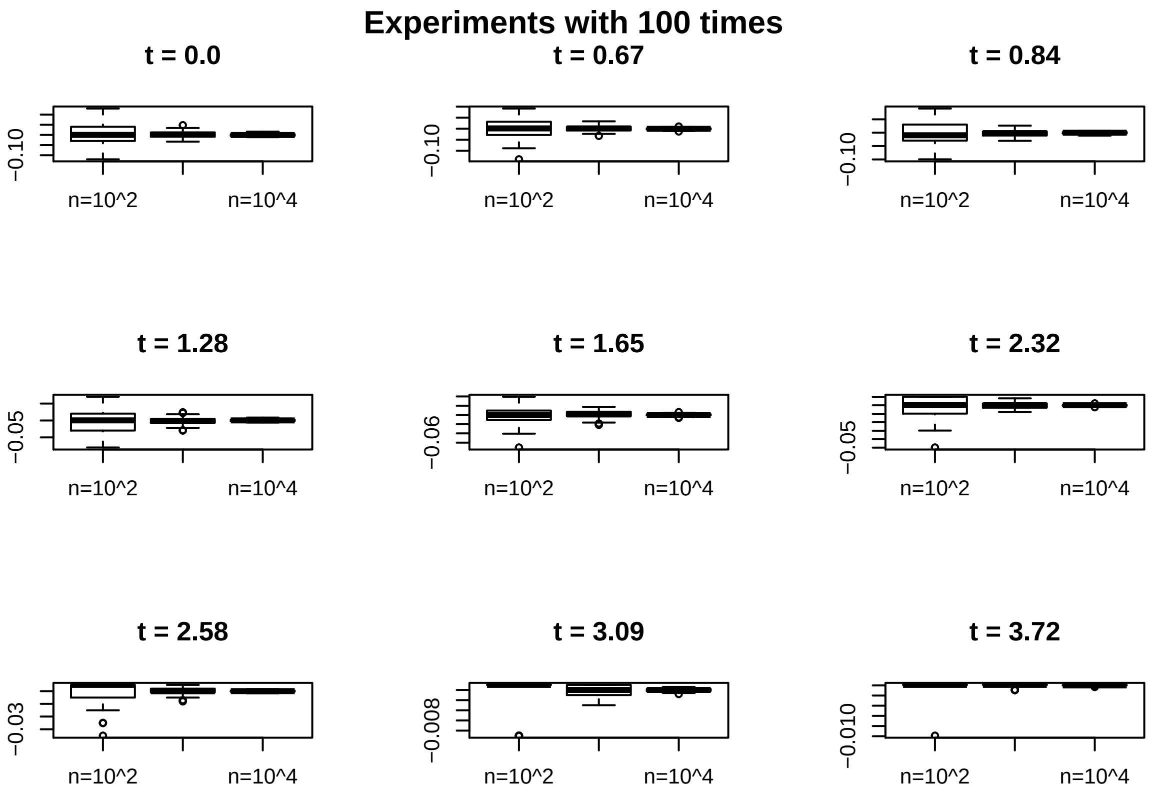
Table for the result:

```
library(knitr)
knitr::kable(data.frame(rbind(experiment[,,1], pnorm(Ts))), format = "markdown",
             row.names = T, col.names = Ts)
```

	0	0.67	0.84	1.28	1.65	2.32	2.58	3.09	3.72
1	0.3800	0.7200000	0.8000000	0.8900000	0.9800000	0.9900000	0.99000	1.0000000	1.0000000
2	0.5090	0.7430000	0.8210000	0.8920000	0.9350000	0.9850000	0.99700	0.9990000	1.0000000
3	0.5038	0.7471000	0.7997000	0.8996000	0.9517000	0.9908000	0.99510	0.9991000	1.0000000
4	0.5000	0.7485711	0.7995458	0.8997274	0.9505285	0.9898296	0.99506	0.9989992	0.9999004

Row 1 is for $n = 10^2$, row 2 is for $n = 10^3$, row 3 is for $n = 10^4$, row 4 is for actual probability. We can see that when n becomes greater and greater, the simulated probability is approximately the actual probability.

Boxplots:



Exercise 3

.Machine$double.xmax: 1.797693e+308

- Since we are using the 64-bit double precision floating point arithmetic, so we have 1 bit of sign, 11 bits of exponent, 52 bits of significand
- First, for sign bit, this bit should take value 0, so $(-1)^0 = 1$, since we want to get the largest number
- Second, for exponent bits, if we fill all 11 bits with 1, then we have $2^0+2^1+2^2+...+2^10 = 2^11-1 = 2047$, but 2047 is reserved for infinity, and also it's normalized by subtracing 1023, so we have $2046-1023 = 1023$
- We fill the significand with all 1, then we $2^{-1} + 2^{-2} + ... + 2^{-52} = 1 - 2^{-52}$, but remember to add 1 before the radix point, we have $1 + 1 - 2^{-52} = 2 - 2^{-52}$
- So if we compute $(2-2^{-52}) \times 2^{1023}$, then we should have $1.797693e+308$, which is .Machine$double.max

.Machine$double.xmin: 2.225074e-308

- First, for sign bit, this bit should take value 0, so $(-1)^0 = 1$, since we want to get a positive number

- Second, for exponent bits, if we fill all 11 bits with 0, then we have 0, but 0 is reserved, and also it's normalized by subtracing 1023, so we have $1 - 1023 = -1022$
- We fill the significand with all 0, then we have 0, but remember to add 1 before the radix point, we have $1 + 0 = 1$
- So if we compute $(1) \times 2^{-1022}$, then we should have $2.225074e - 308$, which is .Machine$double.max

.Machine$double.eps: 2.220446e-16

- This is machine ϵ
- so if we set the last digit in significand to be 1, and remain 0, then we have $1 + 2^{-52} \neq 1$, so 2^{-52} is machine ϵ , it's equal to $2.220446e - 16$

.Machine$double.neg.eps: 1.110223e-16

- Since $1-\epsilon = 1.1111...1110 \times 2^{-1}$, so there is one number between this and 1, which is $1.1111...1111 \times 2^{-1} = 1 - \epsilon/2 \neq 1$, so .Machine$double.neg.eps is $\epsilon/2 = 2^{-53} = 1.110223e - 16$