< STAT-5361 > HW#2-Exercises 2

Hee-Seung, Kim

September 13, 2018

Abstract

Here is exercise 2 for HW 2 in STAT-5361 lecture. We compare the approximation value of standard normal distribution using the Monte Carlo method with a true value generated by pnorm function in R. In order to improve the reliability, we repeat it by 100 times.

Keywords: Template; R Markdown; **bookdown**; **knitr**; **Pandoc**

1 Math Equations

the approximation of the distribution function of $N(0,1)$,

$$\Phi(t) = \int_{-\infty}^{t} \frac{1}{\sqrt{2\pi}} e^{-y^2/2} dy, \tag{1}$$

by the Monte Carlo methods:

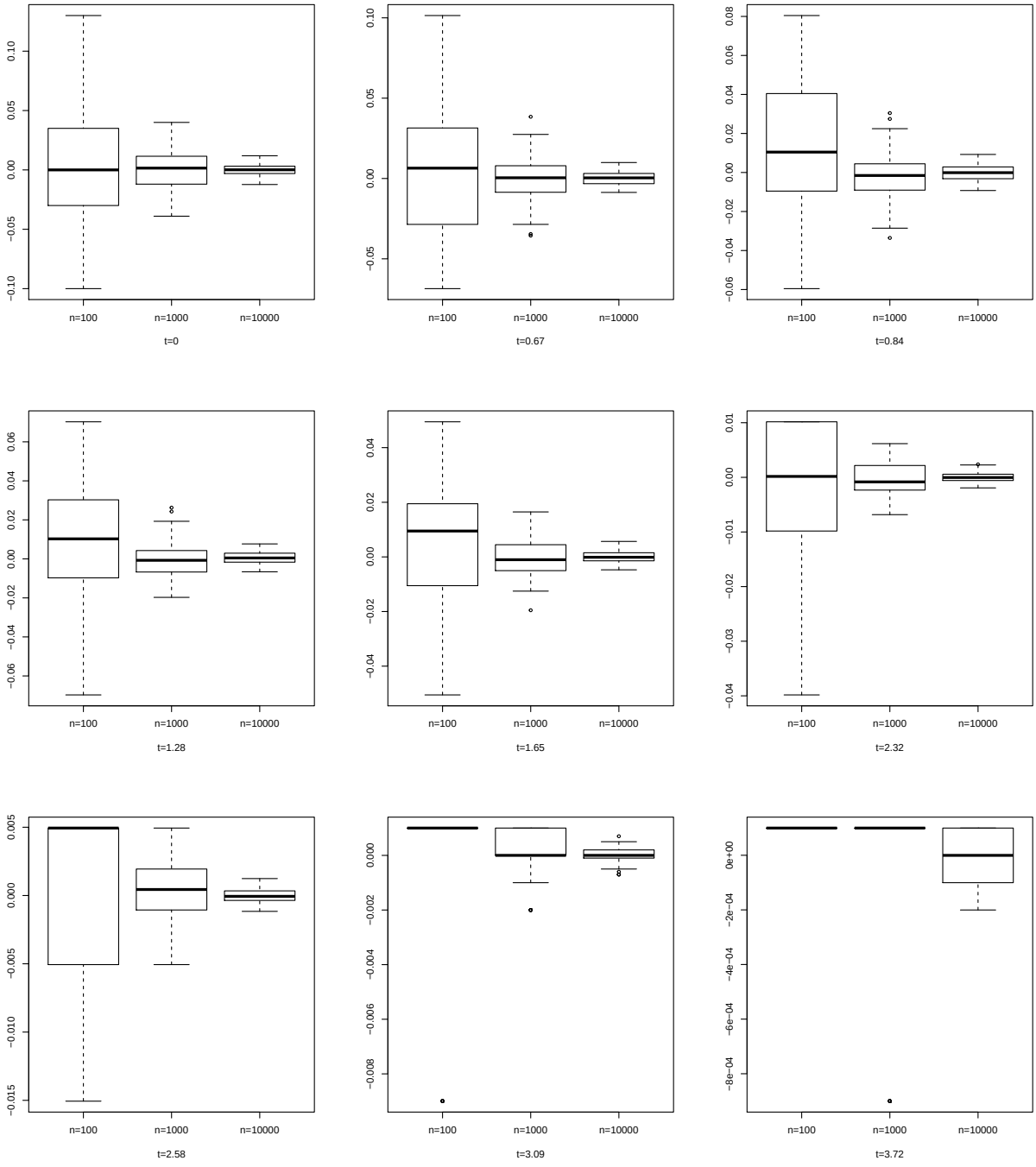
$$\hat{\Phi}(t) = \frac{1}{n} \sum_{i=1}^{n} I(X_i \leq t), \tag{2}$$

where X_i's are iid $N(0,1)$ variables.

Table 1: The approximation result corresponding to different conditions

	t=0	t=0.67	t=0.84	t=1.28	t=1.65	t=2.32	t=2.58	t=3.09	t=3.72
True value	0.5000	0.7486	0.7995	0.8997	0.9505	0.9898	0.9951	0.9990	0.9999
n=100	0.5200	0.8000	0.8300	0.9100	0.9500	0.9700	0.9800	0.9900	1.0000
n=1000	0.4950	0.7350	0.7930	0.8860	0.9580	0.9870	0.9930	1.0000	1.0000
n=10000	0.5076	0.7541	0.8106	0.9081	0.9557	0.9890	0.9951	0.9989	0.9999

2 Table and Figure



3 Code Chunk

Table 2: The approximation result corresponding to different conditions

	t=0	t=0.67	t=0.84	t=1.28	t=1.65	t=2.32	t=2.58	t=3.09	t=3.72
True value	0.5000	0.7486	0.7995	0.8997	0.9505	0.9898	0.9951	0.9990	0.9999
n=100	0.5200	0.8000	0.8300	0.9100	0.9500	0.9700	0.9800	0.9900	1.0000
n=1000	0.4950	0.7350	0.7930	0.8860	0.9580	0.9870	0.9930	1.0000	1.0000
n=10000	0.5076	0.7541	0.8106	0.9081	0.9557	0.9890	0.9951	0.9989	0.9999

```
t<- c(0.0, 0.67, 0.84, 1.28, 1.65, 2.32, 2.58, 3.09, 3.72)
true_value<- pnorm(t, 0, 1)
set.seed(66574565)
n1 = rnorm(100); n2 = rnorm(1000); n3 = rnorm(10000)
An1 = rep(0, length(t))
An2 = rep(0, length(t))
An3 = rep(0, length(t))

for(i in 1:length(t)){
  An1[i] = mean(n1<t[i])
  An2[i] = mean(n2<t[i])
  An3[i] = mean(n3<t[i])
}

Hee=round(rbind(true_value,An1,An2,An3),4)
row.names(Hee)=c("True value","n=100","n=1000","n=10000")
knitr::kable(Hee, col.names = c("t=0","t=0.67","t=0.84","t=1.28","t=1.65","t=2.32","t=2.58","t=
```

```
draw1=matrix(0, 9, 100)
draw2=matrix(0, 9, 100)
draw3=matrix(0, 9, 100)
set.seed(66574565)
re=100
for(i in 1:re){
  n1=rnorm(100)
  for (j in 1:length(t))
    draw1[j,i]=mean(n1<t[j])-true_value[j]
}


re2=1000
for(i in 1:re){
  n2=rnorm(1000)
  for (j in 1:length(t))
    draw2[j,i]=mean(n2<t[j])-true_value[j]
}
```

```r
re3=10000
for(i in 1:re){
  n3=rnorm(10000)
  for (j in 1:length(t))
    draw3[j,i]=mean(n3<t[j])-true_value[j]
}

par(mfrow=c(1,3))
#boxplot(draw1[1,],draw2[1,],draw3[1,], names=c('n=100','n=1000','n=10000'),
#xlab='t=0')
#boxplot(draw1[2,],draw2[2,],draw3[2,], names=c('n=100','n=1000','n=10000'),
#xlab='t=0.67')
#boxplot(draw1[3,],draw2[3,],draw3[3,], names=c('n=100','n=1000','n=10000'),
#xlab='t=0.84')
#boxplot(draw1[4,],draw2[4,],draw3[4,], names=c('n=100','n=1000','n=10000'),
#xlab='t=1.28')
#boxplot(draw1[5,],draw2[5,],draw3[5,], names=c('n=100','n=1000','n=10000'),
#xlab='t=1.65')
#boxplot(draw1[6,],draw2[6,],draw3[6,], names=c('n=100','n=1000','n=10000'),
#xlab='t=2.32')
#boxplot(draw1[7,],draw2[7,],draw3[7,], names=c('n=100','n=1000','n=10000'),
#xlab='t=2.58')
#boxplot(draw1[8,],draw2[8,],draw3[8,], names=c('n=100','n=1000','n=10000'),
#xlab='t=3.09')
#boxplot(draw1[9,],draw2[9,],draw3[9,], names=c('n=100','n=1000','n=10000'),
#xlab='t=3.72')
```

4 Discussion

As we already may know, we can get more closer result to true values by increasing a number of samples. The data repeated by 100 times shows that increasing a number of samples can reduce a fluctuation.