Use the Monte Carlo method to approximate the distribution of $N(0,1)$

Xiaokang Liu

13 September 2018

Abstract

This report includes a small experiment to use the Monte Carlo method to approximate the cumulative distribution function of the standard normal distribution. The results will be displayed in tables and graphs.

Contents

1 Introduction

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

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

by

$$\widehat{\Psi}(t) = \frac{1}{n} \sum_{i=1}^{n} I(X_i \le t), \tag{2}$$

where X_i's are i.i.d. $N(0,1)$ variables. Experiments with the approximation at $n \in \{10^2, 10^3, 10^4\}$ at $t \in \{0, 0.67, 0.84, 1.28, 1.65, 2.32, 2.58, 3.09, 3.72\}$ will be displayed by a table. The experiment will be repeated for 100 times. The bias at all t will be illustrated by boxplots.

2 Implementation and Results

2.1 R code for conducting experiments

```r
n <- c(100, 1000, 10000)
t <- c(0,0.67,0.84,1.28,1.65,2.32,2.58,3.09,3.72)
results <- array(dim = c(100, 3, 9))
for (i in 1:100){
  for (j in 1:3){
    for (k in 1:9){
      results[i,j,k] <- sum(rnorm(n[j])<=t[k])/n[j]
    }
  }
}

# summarize the results from 100 repetitions
sum.results <- matrix(nrow = 4, ncol = 9)
sum.results[1,] <- pnorm(t)
sum.results[2,] <- apply(results[,1,], 2, mean)
sum.results[3,] <- apply(results[,2,], 2, mean)
sum.results[4,] <- apply(results[,3,], 2, mean)
colnames(sum.results) <- c("0.0","0.67","0.84","1.28","1.65","2.32","2.58",
                           "3.09","3.72")
rownames(sum.results) <- c("true","n=100","n=1000","n=10000")
```

2.2 Results

2.2.1 Tables of averaged estimation of probability

The following two tables including the results averaged from 100 repetitions for each situation. By comparing the results of the 2nd, 3rd and 4th row with the 1st row, we can find that in general, larger the sample size, smaller the difference between the approximated probability and the true probability.

```r
# table
knitr::kable(sum.results[,c(1:5)], booktabs = TRUE,
             caption = 'Summary of the experiment(part 1)')
```

```r
knitr::kable(sum.results[,-c(1:5)], booktabs = TRUE,
             caption = 'Summary of the experiment(part 2)')
```

2.2.2 Box plots of bias at all t

The following three boxplots show the bias under different situations. For each plot, we consider three t values. And for each t value, the black boxplot is the one for $n = 100$, the gray one is for

Table 1: Summary of the experiment(part 1)

	0.0	0.67	0.84	1.28	1.65
true	0.500000	0.7485711	0.7995458	0.8997274	0.9505285
n=100	0.499700	0.7497000	0.8038000	0.8980000	0.9490000
n=1000	0.501040	0.7505100	0.8024900	0.9006300	0.9500000
n=10000	0.500005	0.7489920	0.7992290	0.8991250	0.9505230

Table 2: Summary of the experiment(part 2)

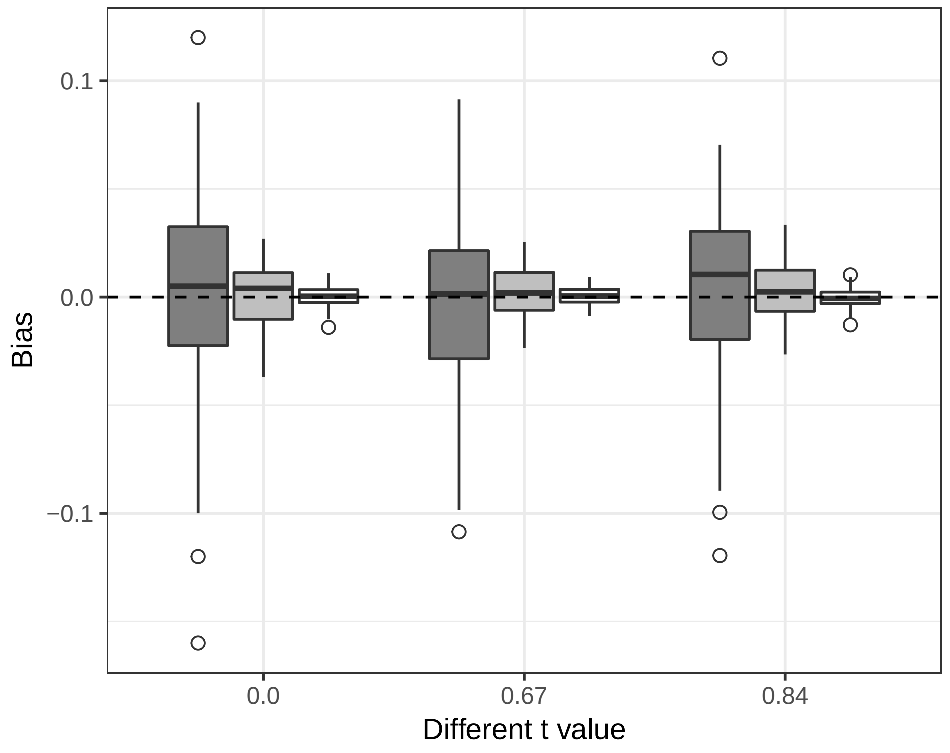
	2.32	2.58	3.09	3.72
true	0.9898296	0.995060	0.9989992	0.9999004
n=100	0.9883000	0.995400	0.9992000	1.0000000
n=1000	0.9897600	0.995120	0.9991200	0.9999300
n=10000	0.9900070	0.995062	0.9990230	0.9999100

$n = 1000$ and the white one is for $n = 10000$. From those boxplots, we can get an overview of the change of the approximate efficiency along with the increasing of sample size and t value.

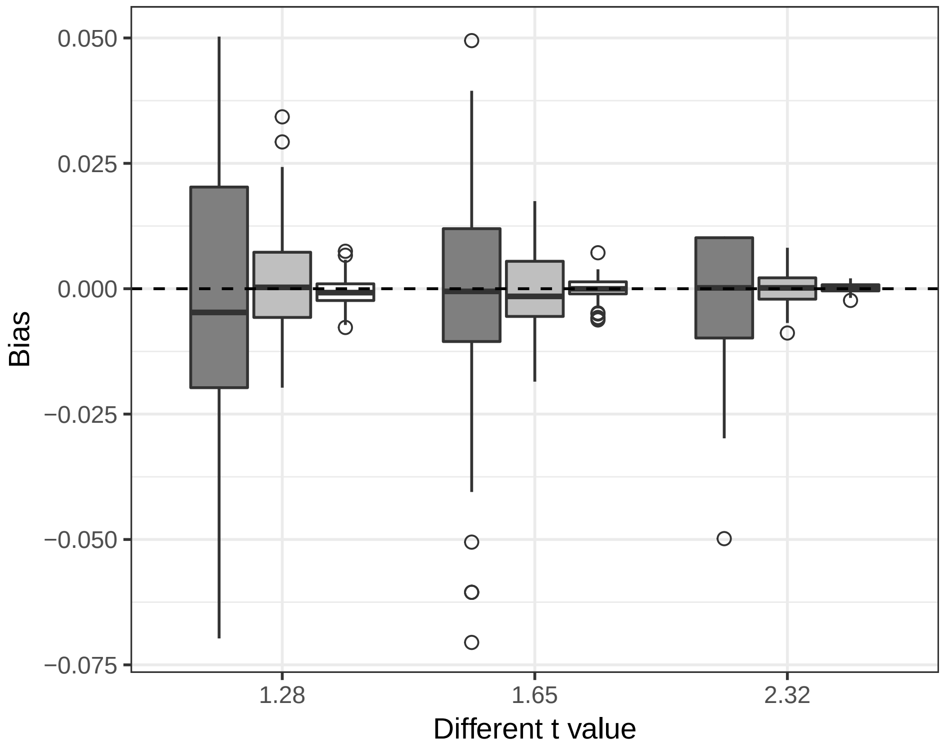
By comparing three boxplots corresponding to three different sample size for one t value, we can find that the estimation bias get closer to 0 when the sample size increases from 100 to 10000. And the distribution of bias become more and more concentrated on 0.

```
# get the bias
bias <- array(dim = c(100, 3, 9))
truep <- t(matrix(rep(pnorm(t),3), nrow = 9, ncol = 3))
for (i in 1:100){
  bias[i,,] <- results[i,,]-truep
}

####### for t=0.0, 0.67, 0.84
prg1 <- vector()
for(a in 1:3){
  for (b in 1:3){
    prg1 <- c(prg1,sort(bias[,a,b]))
  }
}

nprg <- 100
f1 <- rep(c(rep("0.0",nprg),rep("0.67",nprg),rep("0.84",nprg)),3)
f2 <- c(rep(100,nprg*3),rep(1000,nprg*3),rep(10000,nprg*3))

prgdata1 <- data.frame(b=factor(f1),
                      Correlation=factor(f2),
                      PRG=prg1,geom="point")
```

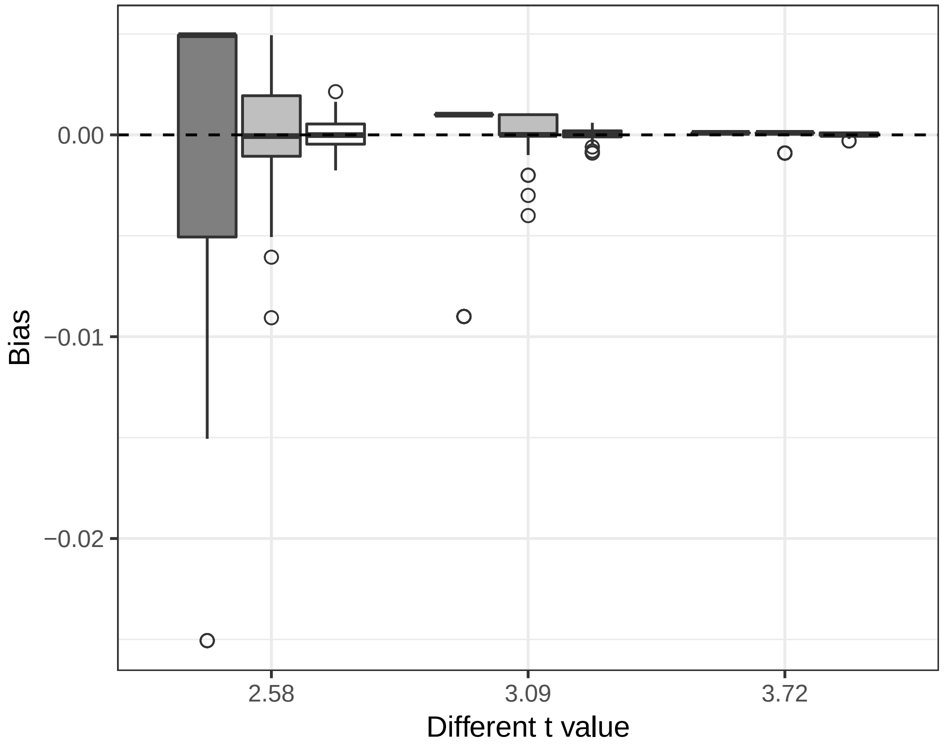
```r
#postscript(paste("1to3.eps",sep=""), width = 4, height = 4,horizontal=FALSE)
ggplot(aes(y = PRG, x = b, fill = Correlation), data = prgdata1) +
  geom_boxplot(notch=FALSE,notchwidth=0.3,outlier.size=2,outlier.shape=1) +
  scale_fill_manual(name = "Correlation",
                    values = c("grey50", "grey75","white"))+
  ylab("Bias") +
  xlab("Different t value")+
  theme_bw()+
  guides(fill=FALSE)+
  geom_hline(aes(yintercept=0), colour="black", linetype="dashed")
```



```r
###### for t=1.28, 1.65, 2.32
prg2 <- vector()
for(a in 1:3){
  for (b in 4:6){
    prg2 <- c(prg2,sort(bias[,a,b]))
  }
}

f3 <- rep(c(rep("1.28",nprg),rep("1.65",nprg),rep("2.32",nprg)),3)
prgdata2 <- data.frame(b=factor(f3),
                      Correlation=factor(f2),
                      PRG=prg2,geom="point")

#postscript(paste("4to5.eps",sep=""), width = 4, height = 4,horizontal=FALSE
ggplot(aes(y = PRG, x = b, fill = Correlation), data = prgdata2) +
  geom_boxplot(notch=FALSE,notchwidth=0.3,outlier.size=2,outlier.shape=1) +
  scale_fill_manual(name = "Correlation",
                    values = c("grey50", "grey75","white"))+
  ylab("Bias") +
  xlab("Different t value")+
  theme_bw()+
  guides(fill=FALSE)+
  geom_hline(aes(yintercept=0), colour="black", linetype="dashed")
```



```
##### for t=2.58, 3.09, 3.72
prg3 <- vector()
for(a in 1:3){
  for (b in 7:9){
    prg3 <- c(prg3,sort(bias[,a,b]))
  }
}
f4 <- rep(c(rep("2.58",nprg),rep("3.09",nprg),rep("3.72",nprg)),3)
prgdata3 <- data.frame(b=factor(f4),
                      Correlation=factor(f2),
                      PRG=prg3,geom="point")

#postscript(paste("7to9.eps",sep=""), width = 4, height = 4,horizontal=FALSE)
ggplot(aes(y = PRG, x = b, fill = Correlation), data = prgdata3) +
  geom_boxplot(notch=FALSE,notchwidth=0.3,outlier.size=2,outlier.shape=1) +
  scale_fill_manual(name = "Correlation",
                    values = c("grey50", "grey75","white"))+
  ylab("Bias") +
  xlab("Different t value")+
  theme_bw()+
  guides(fill=FALSE)+
  geom_hline(aes(yintercept=0), colour="black", linetype="dashed")
```



3 Some important quantities in R

3.1 .Machine$double.xmin

It is the smallest non-zero normalized floating-point number.

```
.Machine$double.xmin
```

```
## [1] 2.225074e-308
```

```
2^(-1022)*(1+0)
```

```
## [1] 2.225074e-308
```

3.2 .Machine$double.xmax

It is the largest normalized floating-point number.

```
.Machine$double.xmax
```

```
## [1] 1.797693e+308
```

```
2^(1023)*(1+(1-2^(-52)))
```

```
## [1] 1.797693e+308
```

3.3 .Machine$double.eps

It is the smallest positive floating-point number x such that $1 + x \neq 1$.

```
.Machine$double.eps
```

```
## [1] 2.220446e-16
```

```
2^(-52)
```

```
## [1] 2.220446e-16
```

3.4 .Machine$double.neg.eps

It is a small positive floating-point number x such that $1 - x \neq 1$.

```
.Machine$double.neg.eps
```

```
## [1] 1.110223e-16
```

```
2^(-53)
```

```
## [1] 1.110223e-16
```