Approximation of Monte Carlo Method

5361 Homework 2

Qinxiao Shi

09/04/2018

Abstract

Exercise 1.2: Consider approximation of standard normal distribution by the Monte Carlo methods.
Exercise 1.3: Explain some double floats.

1 Exercise 1.2

1.1 Functions

I'm going to compare standard Normal distribution:

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

with the Monte Carlo methods, where X_i's are iid $N(0,1)$ variables:

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

, when n is large and goes larger.

Experiment with the approximation at $n \in \{10^2, 10^3, 10^4\}$ at $t \in \{0.0, 0.67, 0.84, 1.28, 1.65, 2.32, 2.58, 3.09, 3.72\}$

```
library(knitr)
library(kableExtra)
library(ggplot2)
library(ggpubr)
```

```
## Loading required package: magrittr
```

```
biasb <- data.frame(matrix(NA, nrow=100, ncol=9))
biasc <- data.frame(matrix(NA, nrow=100, ncol=9))
biasd <- data.frame(matrix(NA, nrow=100, ncol=9))

for (y in 1:100)
{
t <- c(0.0, 0.67, 0.84, 1.28, 1.65, 2.32, 2.58, 3.09, 3.72)
a <- array(data=NA, dim=length(t))

for (i in 1:length(t)){
a[i] <- pnorm(t[i], mean=0, sd=1)
}
t <- c("t=", t)
```

Table 1: The CDF of Standard Normal Distribution

t=	F(x)=
0	0.5
0.67	0.74857110490469
0.84	0.79954580673955
1.28	0.899727432045558
1.65	0.950528531966352
2.32	0.98982956133128
2.58	0.995059984242229
3.09	0.998999217523386
3.72	0.999900388611024

```
a <- c("F(x)=", a)


set.seed(y)
norm100 <- rnorm(100)
norm1000 <- rnorm(1000)
norm10000 <- rnorm(10000)

t <- c(0.0, 0.67, 0.84, 1.28, 1.65, 2.32, 2.58, 3.09, 3.72)
b <- array(data=NA, dim=length(t))
c <- array(data=NA, dim=length(t))
d <- array(data=NA, dim=length(t))

for (i in 1:length(t)){
b[i] <- sum(norm100 <= t[i])/100
c[i] <- sum(norm1000 <= t[i])/1000
d[i] <- sum(norm10000 <= t[i])/10000
}
t <- c("t=", t)
b <- c("Sum= (n=100)", b)
c <- c("Sum= (n=1000)", c)
d <- c("Sum= (n=10000)", d)

for (z in 1:9)
{
  biasb[y,z] <- as.numeric(b[z+1])-as.numeric(a[z+1])
  biasc[y,z] <- as.numeric(c[z+1])-as.numeric(a[z+1])
  biasd[y,z] <- as.numeric(d[z+1])-as.numeric(a[z+1])
}
}

table1 <- array(c(t, a), dim = c(10, 2))
kable(table1, caption = 'The CDF of Standard Normal Distribution', booktabs=T) %>%
  row_spec(1, bold = T, hline_after = T)
```

```
table2 <- array(c(t, b, c, d), dim = c(10, 4))
kable(table2, caption = 'The Summation of Monte Carlo Methods', booktabs=T) %>%
  row_spec(1, bold = T, hline_after = T)
```

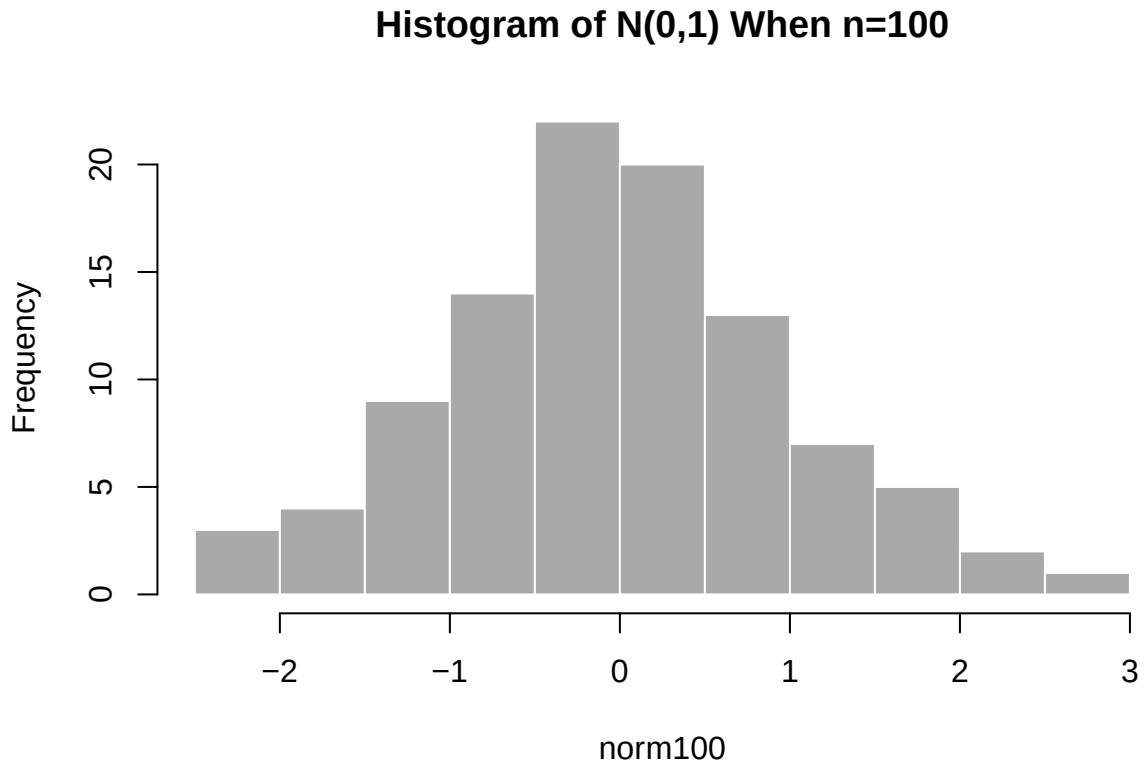
Table 2: The Summation of Monte Carlo Methods

t=	Sum= (n=100)	Sum= (n=1000)	Sum= (n=10000)
0	0.52	0.483	0.5033
0.67	0.75	0.736	0.7504
0.84	0.81	0.794	0.803
1.28	0.87	0.897	0.9044
1.65	0.93	0.95	0.9519
2.32	0.98	0.986	0.99
2.58	0.99	0.994	0.995
3.09	1	0.998	0.9994
3.72	1	1	0.9998

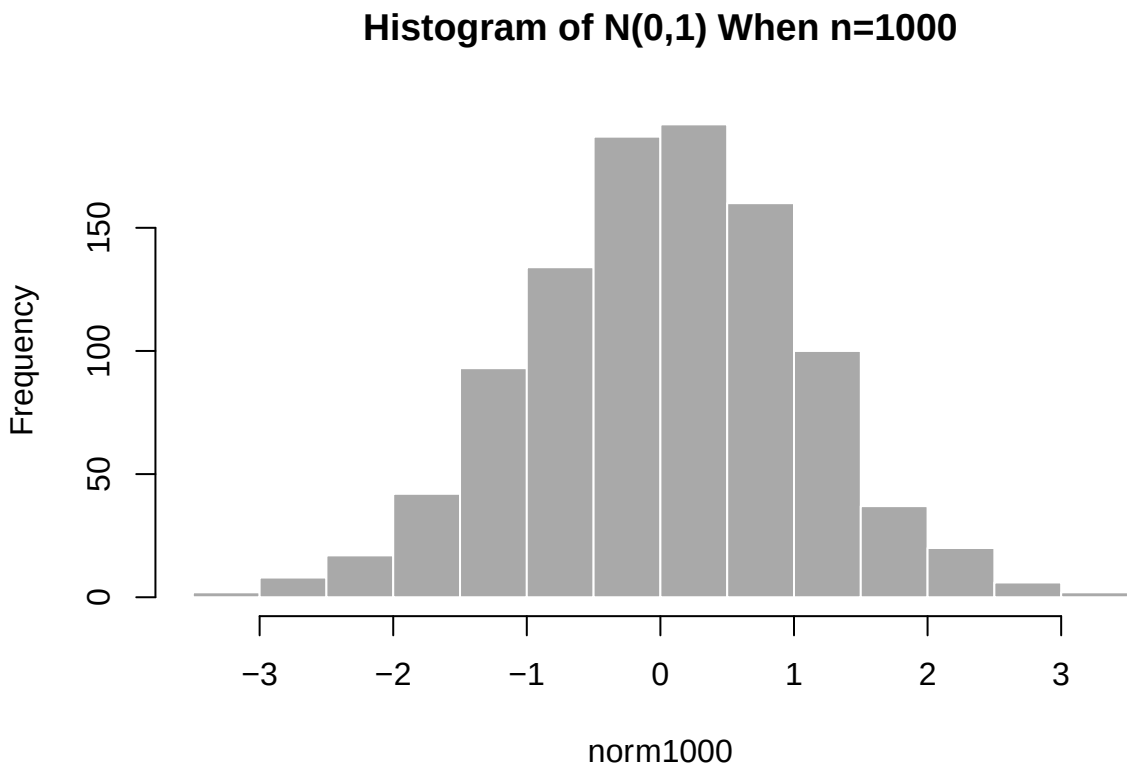
1.2 Figures

It is visual from graphs of two equations:

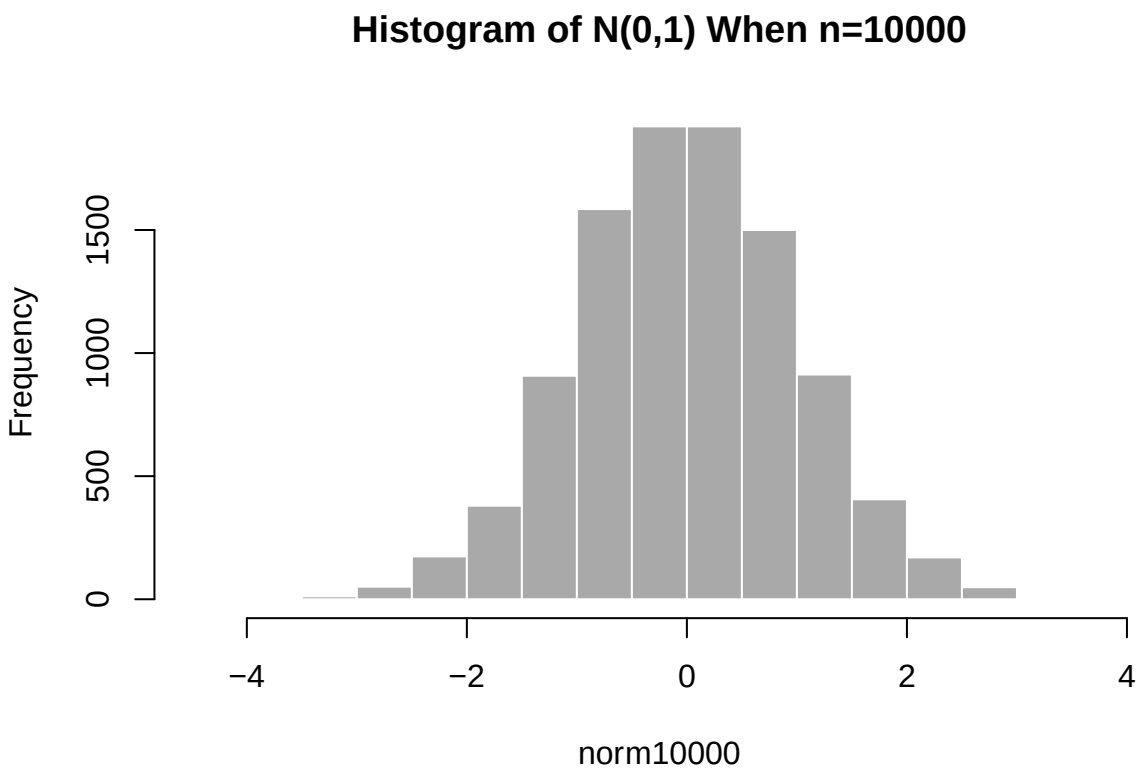
```
hist(norm100, col = 'darkgray', border = 'white', main = "Histogram of N(0,1) When n=100")
```




```
hist(norm1000, col = 'darkgray', border = 'white', main = "Histogram of N(0,1) When n=1000")
```



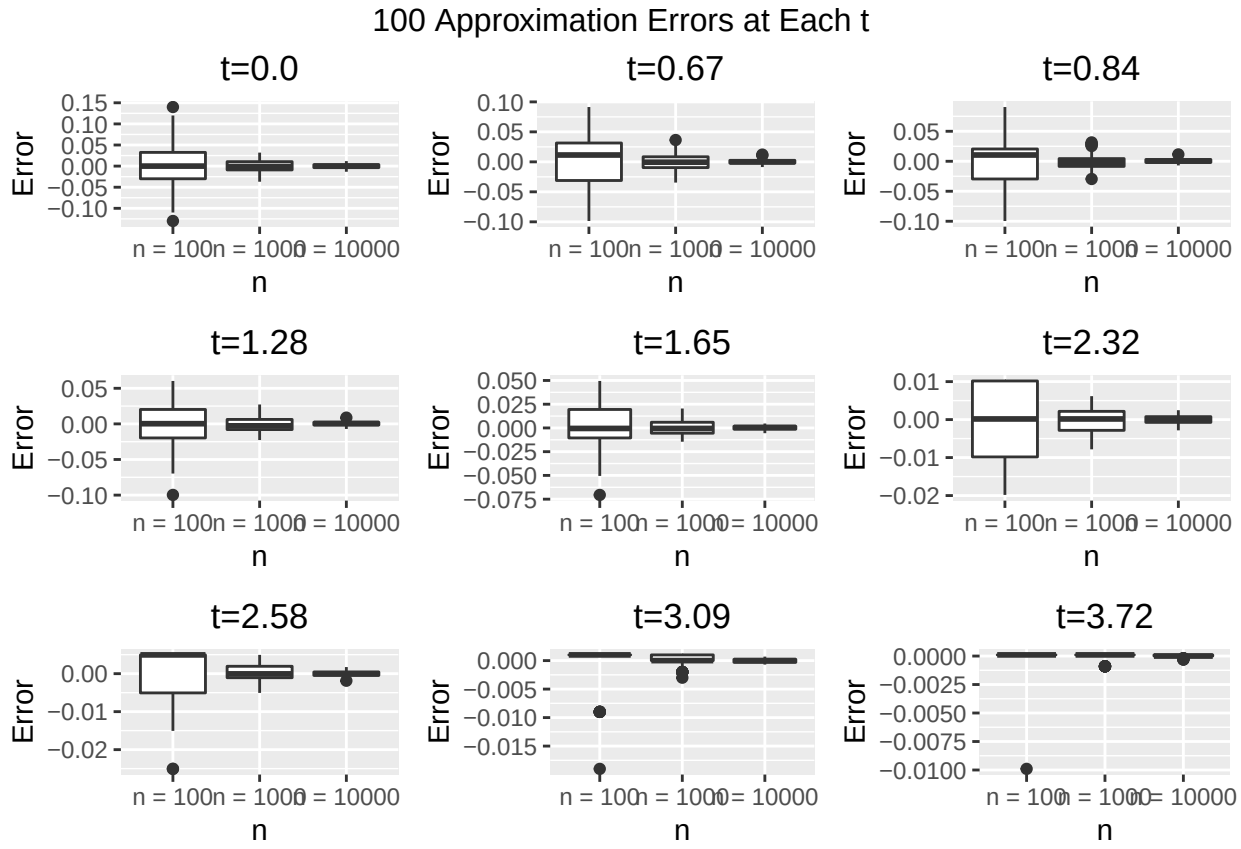

```
hist(norm10000, col = 'darkgray', border = 'white', main = "Histogram of N(0,1) When n=10000")
```



Which is obvious that when n increasing, the distribution of samples random chose from standard normal distribution is more close to standard normal distribution, which is a continues distribution.

```
group <- array(NA, dim=300)
for (x in 1:100){
  group[x] <- "n = 100"
}
for (x in 101:200){
  group[x] <- "n = 1000"
}
for (x in 201:300){
  group[x] <- "n = 10000"
}

norm1 <- c(biasb[,1], biasc[,1], biasd[,1])
table3 <- data.frame(group, norm1)

a <- ggplot(table3, aes(x = group, y = norm1)) +
    geom_boxplot()+ggtitle("t=0.0")+labs(x="n", y="Error")+theme(plot.title = element_text(hjust = 0.5))


norm2 <- c(biasb[,2], biasc[,2], biasd[,2])
table3 <- data.frame(group, norm2)

b <- ggplot(table3, aes(x = group, y = norm2)) +
    geom_boxplot()+ggtitle("t=0.67")+labs(x="n", y="Error")+theme(plot.title = element_text(hjust = 0.5))
```

```r
norm3 <- c(biasb[,3], biasc[,3], biasd[,3])
table3 <- data.frame(group, norm3)

c <- ggplot(table3, aes(x = group, y = norm3)) +
    geom_boxplot()+ggtitle("t=0.84")+labs(x="n", y="Error")+theme(plot.title = element_text(hjust = 0.5)


norm4 <- c(biasb[,4], biasc[,4], biasd[,4])
table3 <- data.frame(group, norm4)

d <- ggplot(table3, aes(x = group, y = norm4)) +
    geom_boxplot()+ggtitle("t=1.28")+labs(x="n", y="Error")+theme(plot.title = element_text(hjust = 0.5)


norm5 <- c(biasb[,5], biasc[,5], biasd[,5])
table3 <- data.frame(group, norm5)

e <- ggplot(table3, aes(x = group, y = norm5)) +
    geom_boxplot()+ggtitle("t=1.65")+labs(x="n", y="Error")+theme(plot.title = element_text(hjust = 0.5)


norm6 <- c(biasb[,6], biasc[,6], biasd[,6])
table3 <- data.frame(group, norm6)

f <- ggplot(table3, aes(x = group, y = norm6)) +
    geom_boxplot()+ggtitle("t=2.32")+labs(x="n", y="Error")+theme(plot.title = element_text(hjust = 0.5)


norm7 <- c(biasb[,7], biasc[,7], biasd[,7])
table3 <- data.frame(group, norm7)

g <- ggplot(table3, aes(x = group, y = norm7)) +
    geom_boxplot()+ggtitle("t=2.58")+labs(x="n", y="Error")+theme(plot.title = element_text(hjust = 0.5)


norm8 <- c(biasb[,8], biasc[,8], biasd[,8])
table3 <- data.frame(group, norm8)

h <- ggplot(table3, aes(x = group, y = norm8)) +
    geom_boxplot()+ggtitle("t=3.09")+labs(x="n", y="Error")+theme(plot.title = element_text(hjust = 0.5)


norm9 <- c(biasb[,9], biasc[,9], biasd[,9])
table3 <- data.frame(group, norm9)

i <- ggplot(table3, aes(x = group, y = norm9)) +
    geom_boxplot()+ggtitle("t=3.72")+labs(x="n", y="Error")+theme(plot.title = element_text(hjust = 0.5)

gridExtra::grid.arrange(a, b, c, d, e, f, g, h, i,
                        nrow=3, ncol=3,
                        top="100 Approximation Errors at Each t")
```



The box plots show that when n goes larger and larger, the error of Monte Carlo methods approaches to 0, which means when sample size increasing, the Monte Carlo methods can more accurately simulate $N(0,1)$.

Since the highly similarity of two distributions, cdf of $N(0,1)$ can be simulated by Monte Carlo method in calculation.

2 Exercise 1.3

The double-precision floating-point format is is a computer number format, usually occupying 64 bits in computer memory.

$$(-1)^{sign}(1+\sum_{i=1}^{52} b_{52-i}2^{-i}) \times 2^{e-1023}$$

2.1 .Machine$double.xmax

Which is the largest normalized floating-point number. In decimal system, it can be denoted as

$$(-1)^0(1+(1-2^{-52}))2^{1023}$$

In binary system, it can be denoted as $0\ 11111111110\ 1111111111111111111111111111111111111111111111111111_2$

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

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

2.2 .Machine$double.xmin

Which is the smallest non-vanishing normalized floating-point power of the radix. In decimal system, it can be denoted as

$$(-1)^0(1+1)2^{-1023}$$

In binary system, it can be denoted as $0\ 00000000001\ 0000000000000000000000000000000000000000000000000000_2$

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

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

2.3 .Machine$double.eps

Which is the smallest positive floating-point number x such that $1 + x\ != 1$. In decimal system, it can be denoted as

$$2^{-52}$$

In binary system, it can be denoted as 00000000001_2

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

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

2.4 .Machine$double.neg.eps

Which is a small positive floating-point number x such that $1 - x\ != 1$. In decimal system, it can be denoted as

$$2^{-53}$$

In binary system, it can be denoted as 00000000000_2

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

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