# Statistical Computing - HW2

*Yaqiong Yao*

*9/13/2018*

**Abstract**

We use this homework to familiar with Rmarkdown and also to learn how computers process numbers, especially the double floating numbers.

## Question 2

Use Monte Carlo method to approximate the distribution function of

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

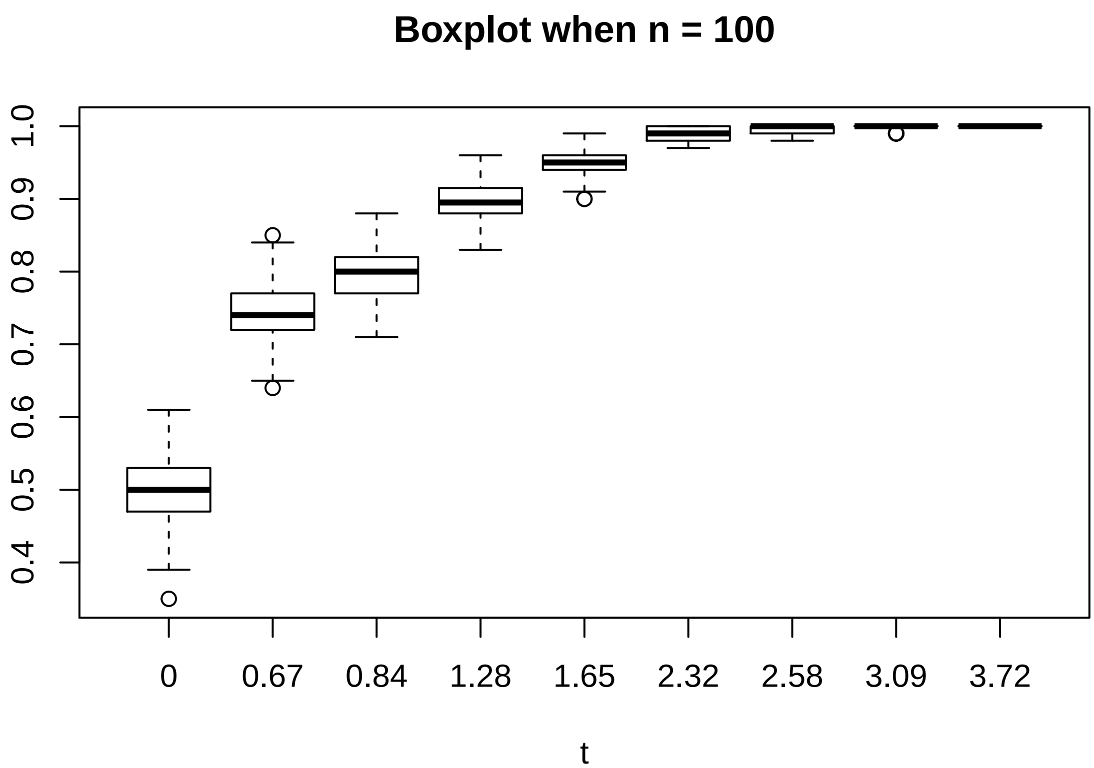
```r
n <- c(1e2, 1e3, 1e4)
t <- c(0.0, 0.67, 0.84, 1.28, 1.65, 2.32, 2.58, 3.09, 3.72)
itr <- 100
set.seed(1)
Phi <- matrix(NA, nrow = length(n), ncol = length(t))
for (ni in seq_along(n)){
  X_ni <- rnorm(n[ni], mean = 0, sd = 1)
  for (ti in seq_along(t)) {
    Phi[ni, ti] <- sum(X_ni <= t[ti])/n[ni]
  }
}
Phi <- rbind(pnorm(t, mean = 0, sd = 1, options(digits = 4)), Phi)
colnames(Phi) <- t
rownames(Phi) <- c("True Value", "100", "1000", "10000")
knitr::kable(Phi)
```

| | 0 | 0.67 | 0.84 | 1.28 | 1.65 | 2.32 | 2.58 | 3.09 | 3.72 |
|---|---|---|---|---|---|---|---|---|---|
| True Value | 0.5000 | 0.7486 | 0.7995 | 0.8997 | 0.9505 | 0.9898 | 0.9951 | 0.9990 | 0.9999 |
| 100 | 0.4600 | 0.7400 | 0.8200 | 0.9100 | 0.9700 | 0.9900 | 1.0000 | 1.0000 | 1.0000 |
| 1000 | 0.5210 | 0.7390 | 0.7940 | 0.8920 | 0.9360 | 0.9900 | 0.9960 | 0.9990 | 0.9990 |
| 10000 | 0.5056 | 0.7496 | 0.7937 | 0.8967 | 0.9521 | 0.9910 | 0.9960 | 0.9994 | 1.0000 |

From the table, we know that the monte carlo method works well. The estimated distribution is approximated to the true distribution.

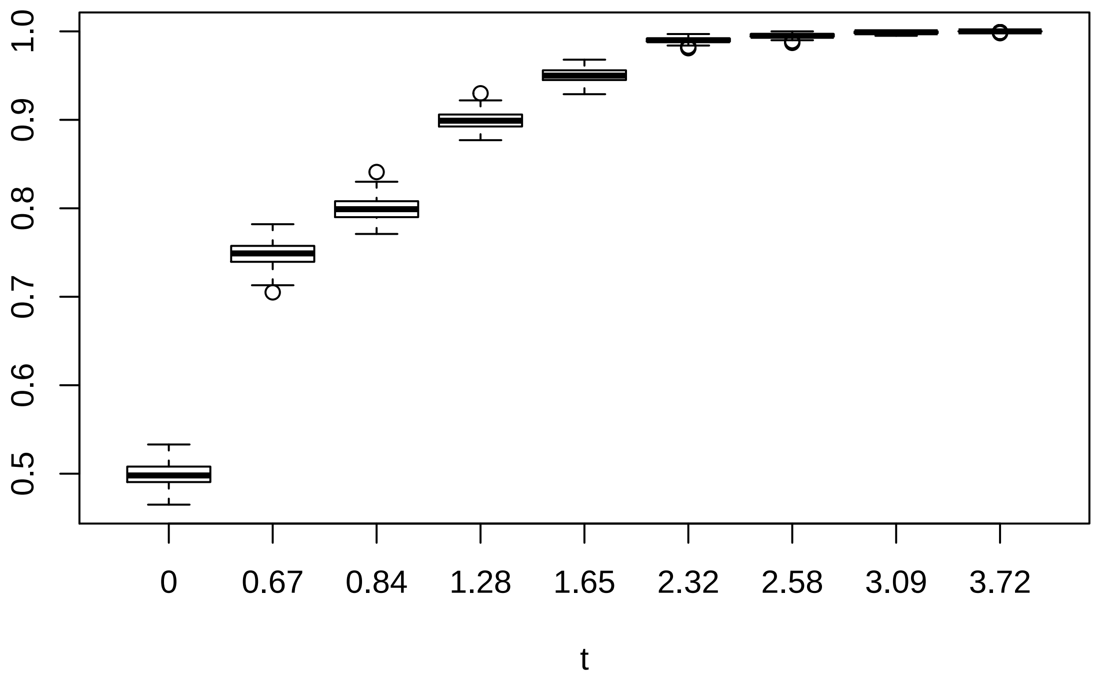
```r
# iteration for 100 times
set.seed(12)
Phi <- list()
for (ni in seq_along(n)) {
  Phii <- matrix(NA, nrow = itr, ncol = length(t))
  for (i in 1:itr) {
    X_ni <- rnorm(n[ni], mean = 0, sd = 1)
    for (ti in seq_along(t)) {
      Phii[i, ti] <- sum(X_ni <= t[ti])/n[ni]
```

```r
    }
  }
  Phi[[ni]] <- Phii
}
# Boxplot
boxplot(Phi[[1]], names = t, xlab = "t", main = "Boxplot when n = 100")
```



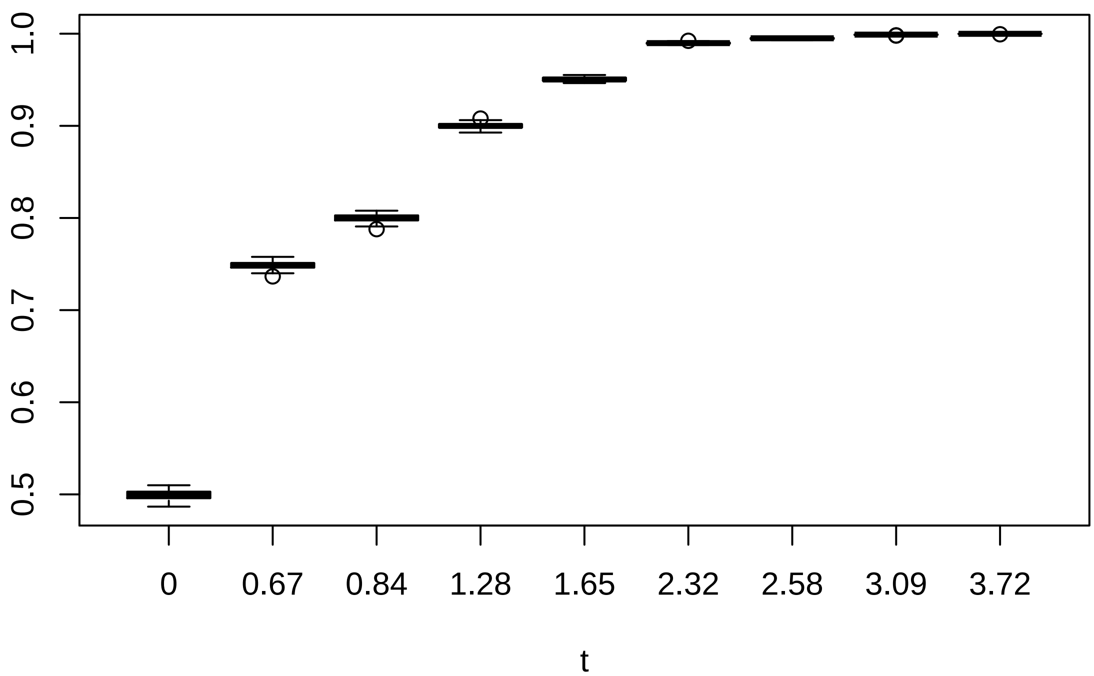

```r
boxplot(Phi[[2]], names = t, xlab = "t", main = "Boxplot when n = 1000")
```

**Boxplot when n = 1000**




```r
boxplot(Phi[[3]], names = t, xlab = "t", main = "Boxplot when n = 10000")
```

**Boxplot when n = 10000**



# Question 3

`.Machine$double.xmax`: The largest normalized floating-point number. It can be expressed by $(-1)^0(1+\sum_{i=1}^{52} 2^{-i}) \times 2^{2046-1023}$. The first 10 digits of exponent part are 1 and 52 bits of fraction are all 1.

```
u <- 0
for(i in 1:52) u <- u+2^(-i)
(1+u)*2^(1023) == .Machine$double.xmax
```

```
## [1] TRUE
```

`.Machine$double.xmin`: The smallest non-zero normalized floating-point number. It can be expressed as $(-1)^0 2^{1-1023}$. Only last bit of exponent is 1 and all 52 bits of fraction are 0.

```
2^{1-1023} == .Machine$double.xmin
```

```
## [1] TRUE
```

`.Machine$double.eps`: The smallest positive floating-point number $x$ such that $1+x \neq 1$. It equals to $2^{-52}$.

```
2^(-52) == .Machine$double.eps
```

```
## [1] TRUE
```

`.Machine$double.neg.eps`: A small positve floating-point number $x$ such that $1-x \neq 1$. It equals to $2^{-53}$.

```
2^(-53) == .Machine$double.neg.eps
```

```
## [1] TRUE
```