STAT 4060J: Homework1

Boqian Wang

2023-09-16

```r
knitr::opts_chunk$set(echo = TRUE)
# if you are using libraries, it's good practice to load them here
library(ggplot2)
```

```
## Warning: 'ggplot2'R4.3.1
```

Question 1

1a. We first check the dimensionality: $\nabla_x f(x)$ is $n \times 1$ and $\nabla^2 f(x)$ is $n \times n$. Then, we treat A, x and b as scalars instead of vectors or matrices. Therefore, $\nabla_x f(x) = \frac{d}{dx}(\frac{1}{2}Ax^2 + bx) = Ax + b$, and $\nabla^2 f(x) = \frac{d^2}{dx^2}(\frac{1}{2}Ax^2 + bx) = A$. The dimensionality of A, x and b are $n \times n$, $n \times 1$ and $n \times 1$ respectively. Therefore, in terms of $Ax + b$, the dimensionality should be $n \times 1$, which matches the dimensionality of $\nabla_x f(x)$. Similarly, the dimensionality of A is $n \times n$, which also matches the dimensionality of $\nabla^2 f(x)$. To summarize, $\nabla_x f(x) = Ax + b$ and $\nabla^2 f(x) = A$.

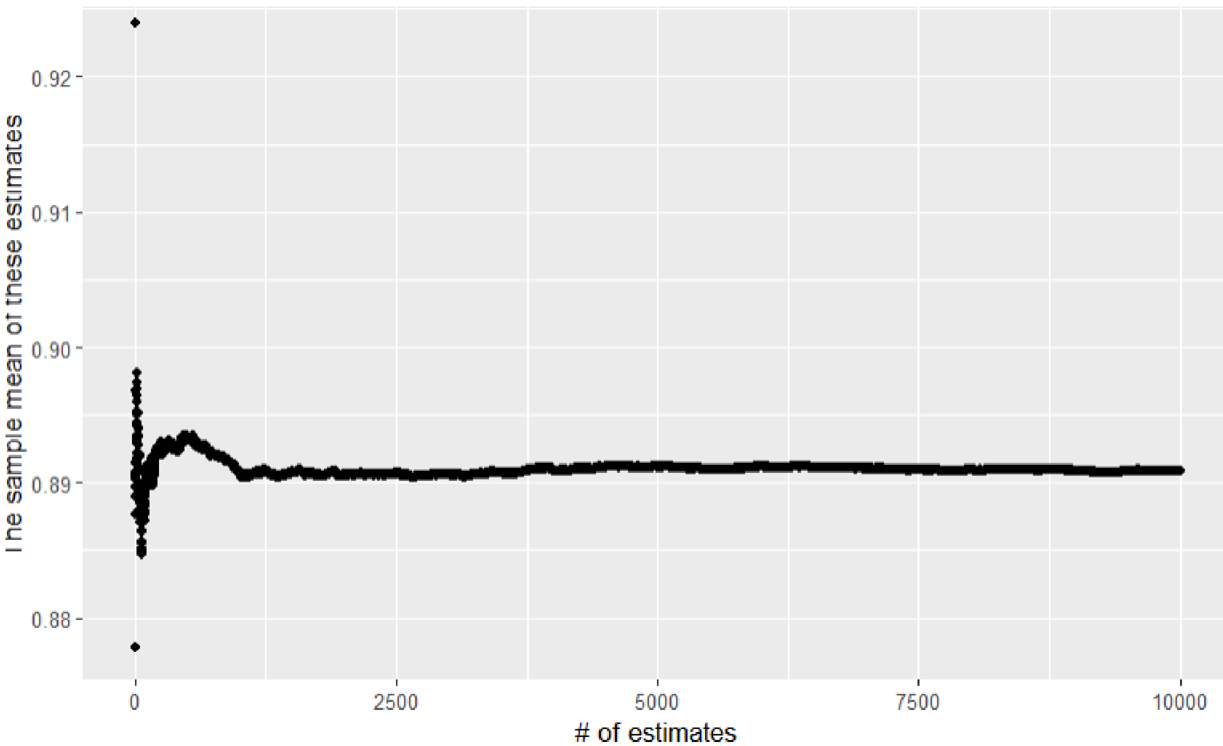
1b. We treat the matrix differentiation as scalar differntiation. According to the chain rule, $\frac{d}{dx}(g(h(x))) = \frac{dg(h)}{dh} \cdot \frac{dh(x)}{dx}$. After that, we note that g is a function from R to R, so $\frac{dg(h)}{dh}$ is a scalar. h is a function from R^n to R, so $\frac{dh(x)}{dx}$ is given by $\nabla_x h(x)$ in the matrix form. So in the matrix form, $\nabla_x f(x) = \nabla_x h(x) \cdot \frac{dg(h)}{dh} = \begin{bmatrix} \frac{\partial}{\partial x_1}(f(x)) \cdot \frac{dg(h)}{dh} \\ \frac{\partial}{\partial x_2}(f(x)) \cdot \frac{dg(h)}{dh} \\ \vdots \\ \frac{\partial}{\partial x_n}(f(x)) \cdot \frac{dg(h)}{dh} \end{bmatrix}$. After checking the dimensionality, they match.

1c. $\nabla_x f(x) = \nabla_x h(x) \cdot \frac{dg(h)}{dh} = a \cdot \frac{dg(h)}{dh}$. The dimensionality matches because $\nabla_x f(x)$ has a dimensionality of $n \times 1$, a also has a dimensionality of $n \times 1$ and $\frac{dg(h)}{dh}$ is a scalar. Considering $\nabla^2 f(x)$, we can also apply the chain rule of the scalar differentiation: $\frac{d^2}{dx^2}[g(h(x))] = \frac{d}{dx}(\frac{dg}{dh}\frac{dh}{dx}) = \frac{d^2g}{dh^2}(\frac{dh}{dx})^2 + \frac{dg}{dh}\frac{d^2h}{dx^2}$. As for matrices, $\nabla^2 f(x) = \frac{d^2g}{dh^2}(\nabla_x h(x))^2 + \frac{dg}{dh}\nabla^2 h(x)$. The dimensionality matches because $\frac{d^2g}{dh^2}$ and $\frac{dg}{dh}$ are scalars, while $(\nabla_x h(x))^2$ and $\nabla^2 h(x)$ are $n \times n$ matrices. After that, we plug in $h(x) = a^T x$, so the final solution is given by $\nabla_x f(x) = a \cdot g'(a^T x)$ and $\nabla^2 f(x) = g''(a^T x) \cdot a^T a$.

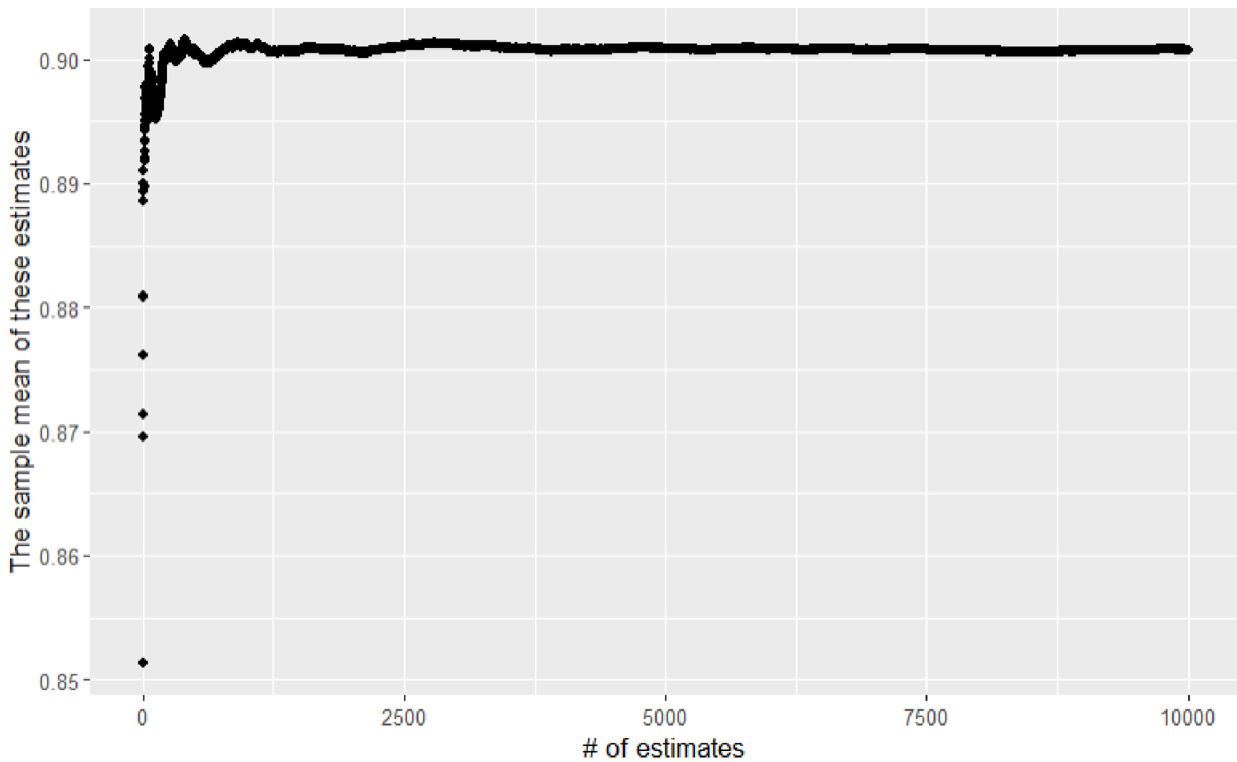
Question 2

```r
# 2a
function2a <- function(x){
  return(sort(x)[90])
}
# 2b
function2b <- function(x){
  return(sort(x)[91])
```

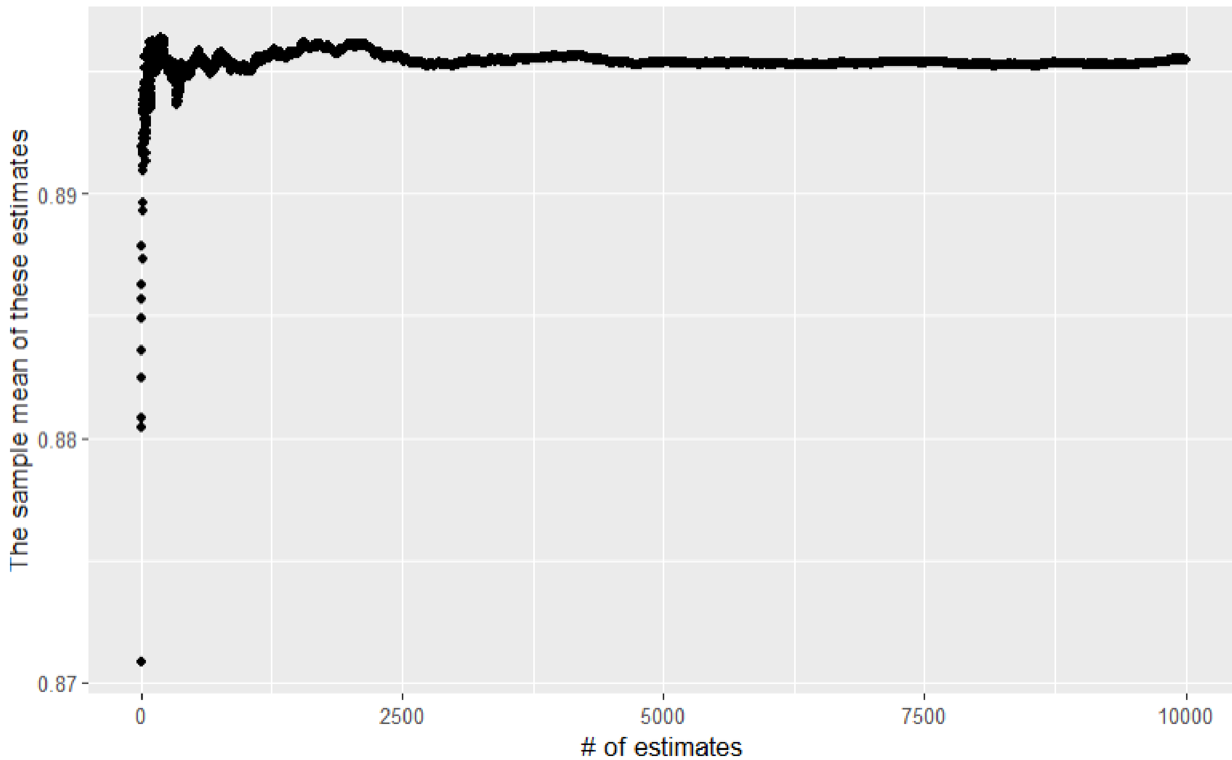
```r
}
# 2c
function2c <- function(x){
  return((function2a(x)+function2b(x))/2)
}
# 2d(i)
# For function 2a:
vectora <- vector(mode="numeric",length=10000)
mean <- 0
for(i in 1:10000){
  temp <- runif(100)
  mean <- (mean*(i-1)+function2a(temp))/i
  vectora[i] <- mean
}
dataa <- data.frame(x=1:10000,y=vectora)
ggplot(dataa, aes(x = x, y = y)) + geom_point()+xlab("# of estimates")+ylab("The sample mean of these e
```



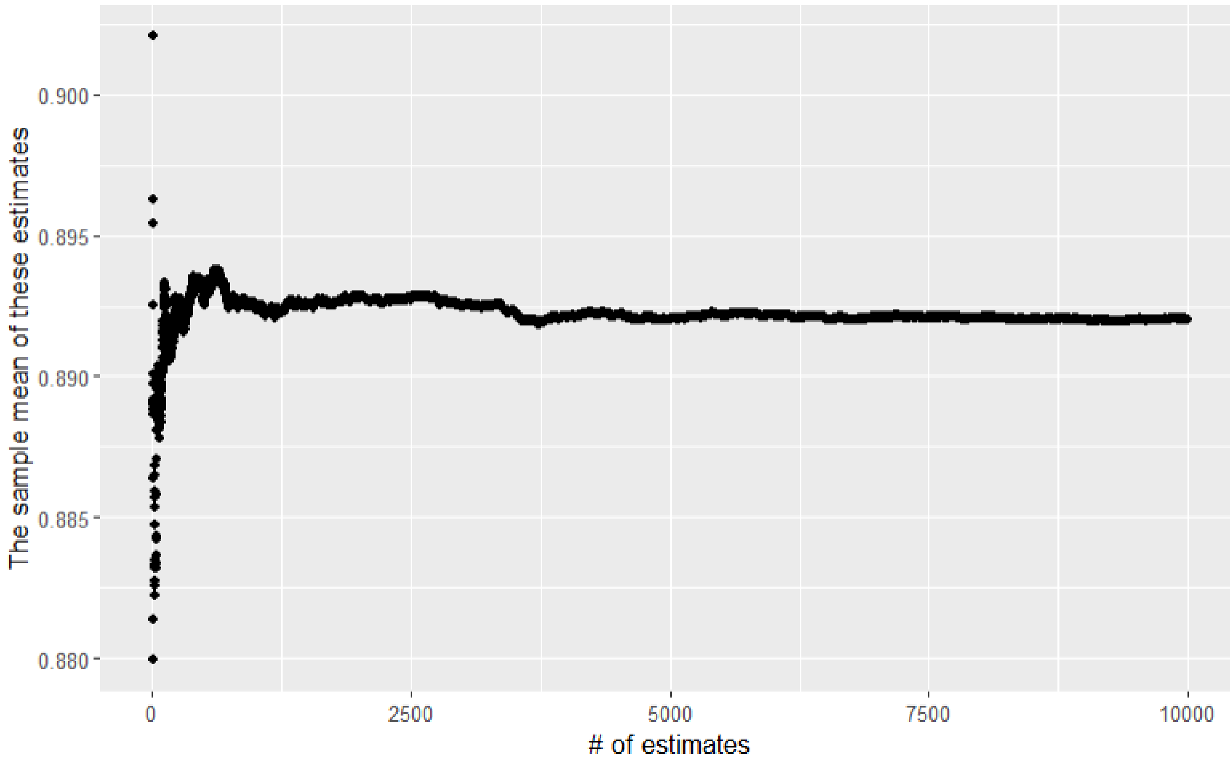

```r
# For function 2b:
vectorb <- vector(mode="numeric",length=10000)
mean <- 0
for(i in 1:10000){
  temp <- runif(100)
  mean <- (mean*(i-1)+function2b(temp))/i
  vectorb[i] <- mean
}
datab <- data.frame(x=1:10000,y=vectorb)
ggplot(datab, aes(x = x, y = y)) + geom_point()+xlab("# of estimates")+ylab("The sample mean of these e
```




```
# For function 2c:
vectorc <- vector(mode="numeric",length=10000)
mean <- 0
for(i in 1:10000){
  temp <- runif(100)
  mean <- (mean*(i-1)+function2c(temp))/i
  vectorc[i] <- mean
}
datac <- data.frame(x=1:10000,y=vectorc)
ggplot(datac, aes(x = x, y = y)) + geom_point()+xlab("# of estimates")+ylab("The sample mean of these e
```




```r
# For function quantile(x,0.9)
vectord <- vector(mode="numeric",length=10000)
mean <- 0
for(i in 1:10000){
  temp <- runif(100)
  mean <- (mean*(i-1)+quantile(temp,0.9))/i
  vectord[i] <- mean
}
datad <- data.frame(x=1:10000,y=vectord)
ggplot(datad, aes(x = x, y = y)) + geom_point()+xlab("# of estimates")+ylab("The sample mean of these es
```



2e. I think a is the best candidate, as function a owns the closest sample mean of these estimates to function d, compared to function b and function c.

Question 3

```
# 3a
pi2 <- function(x){
  sum <- 0
  for(i in 1:x){
    sum <- sum + i^(-2)
  }
  return(sqrt(sum*6))
}
print(pi2(1))        # j = 0
```

```
## [1] 2.44949
```

```
print(pi2(10))       # j = 1
```

```
## [1] 3.049362
```

```
print(pi2(100))      # j = 2
```

```
## [1] 3.132077
```

```
print(pi2(1000))     # j = 3
```

```
## [1] 3.140638
```

```
print(pi2(10000))    # j = 4
```

```
## [1] 3.141497
```

```r
print(pi2(100000))  # j = 5
```

```
## [1] 3.141583
```

```r
print(pi2(1000000)) # j = 6
```

```
## [1] 3.141592
```

```r
# 3b
pi3 <- function(x){
  sum <- 0
  for(i in 1:x){
    temp1 <- runif(1, -1, 1)
    temp2 <- runif(1, -1, 1)
    if(temp1^2 + temp2^2 < 1){
      sum <- sum + 1
    }
  }
  return(4*sum/x)
}
print(pi3(1))       # j = 0
```

```
## [1] 4
```

```r
print(pi3(10))      # j = 1
```

```
## [1] 3.2
```

```r
print(pi3(100))     # j = 2
```

```
## [1] 3.04
```

```r
print(pi3(1000))    # j = 3
```

```
## [1] 3.06
```

```r
print(pi3(10000))   # j = 4
```

```
## [1] 3.1536
```

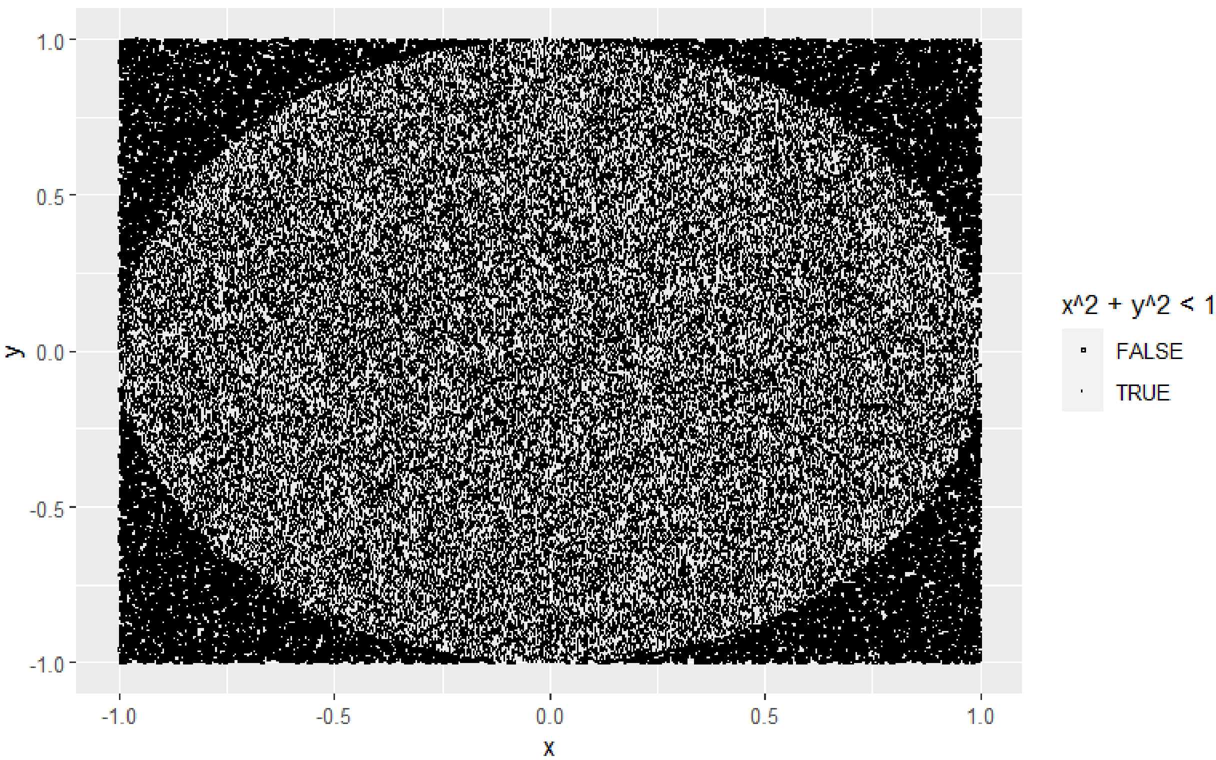
```r
print(pi3(100000))  # j = 5
```

```
## [1] 3.13628
```

```r
print(pi3(1000000)) # j = 6
```

```
## [1] 3.139836
```

```r
# 3c
# When j = 6, my R studio gets stuck. So my works are based on j = 5.
vector1 <- vector(mode="numeric",length=100000)
vector2 <- vector(mode="numeric",length=100000)
for(i in 1:100000){
  vector1[i] <- runif(1, -1, 1)
  vector2[i] <- runif(1, -1, 1)
}
data = data.frame(x=vector1,y=vector2)
ggplot(data, aes(x = x, y = y)) + geom_point(aes(x = x, y = y, shape = x^2+y^2<1), size = 0.1) +scale_s
```


1.0
0.5
0.0
-0.5
-1.0
y
-1.0
-0.5
0.0
0.5
1.0
x
x^2 + y^2 < 1
FALSE
TRUE