Homework #4

J. Hamski

10/13/2016

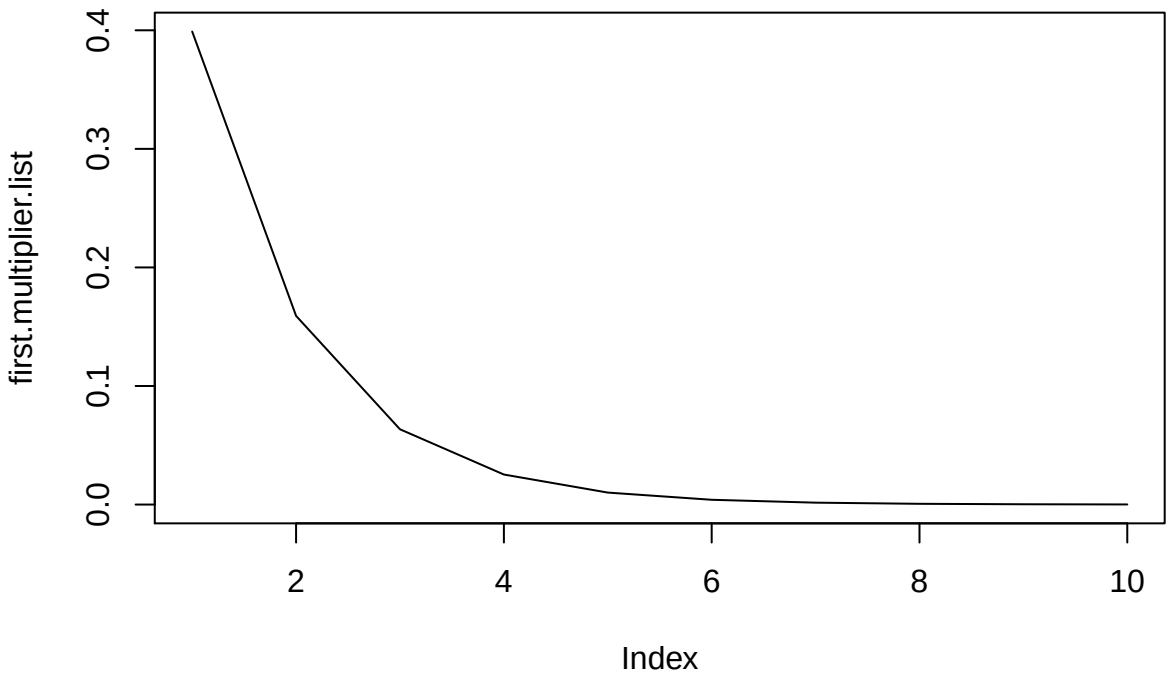
```
library(ggplot2)
library(magrittr)
```

The cost function

The multiplier decreases as D increases.

```
first.multiplier <- function(D){
  return(1/((2*pi)^(D/2)))
}
```

```
D.seq <- seq(1, from=1, to=10)

first.multiplier.list <- sapply(D.seq, FUN=first.multiplier)

plot(first.multiplier.list,  type = "l")
```



Testing the exponent function.

```
exponent.function <- function(x){
  return((-1/2)*t(x)%*%x)
}

D <- 10
```

```
x <- as.matrix(runif(D, min = -5, max = 5), nrow=1, ncol=D)

exponent.function(x)
```

```
##           [,1]
## [1,] -46.12495
```

These don't really need to be broken out into different functions, but I found it helpful to try different inputs and see how the components of the function behave.

a) Crude Monte Carlo
First, create a function to evaluate c(x) at N samples in D dimensions (i.e., a DxN matrix).

```
cost.function.crude <- function(x, D){
  first.multiplier.sim <- first.multiplier(D)
  exponent.sim <- exponent.function(x)
  return(first.multiplier.sim * exp(exponent.sim))
}
```

```
n <- 100
D <- 1
x <- as.matrix(runif(D*n, min = -5, max = 5), nrow=n, ncol=D)
```

```
cost.function.crude.apply <- function(x, D){
  return(mean(apply(x, 1, FUN = cost.function.crude, D=D)))
}
cost.function.crude.apply(x, D)
```

```
## [1] 0.1190923
```

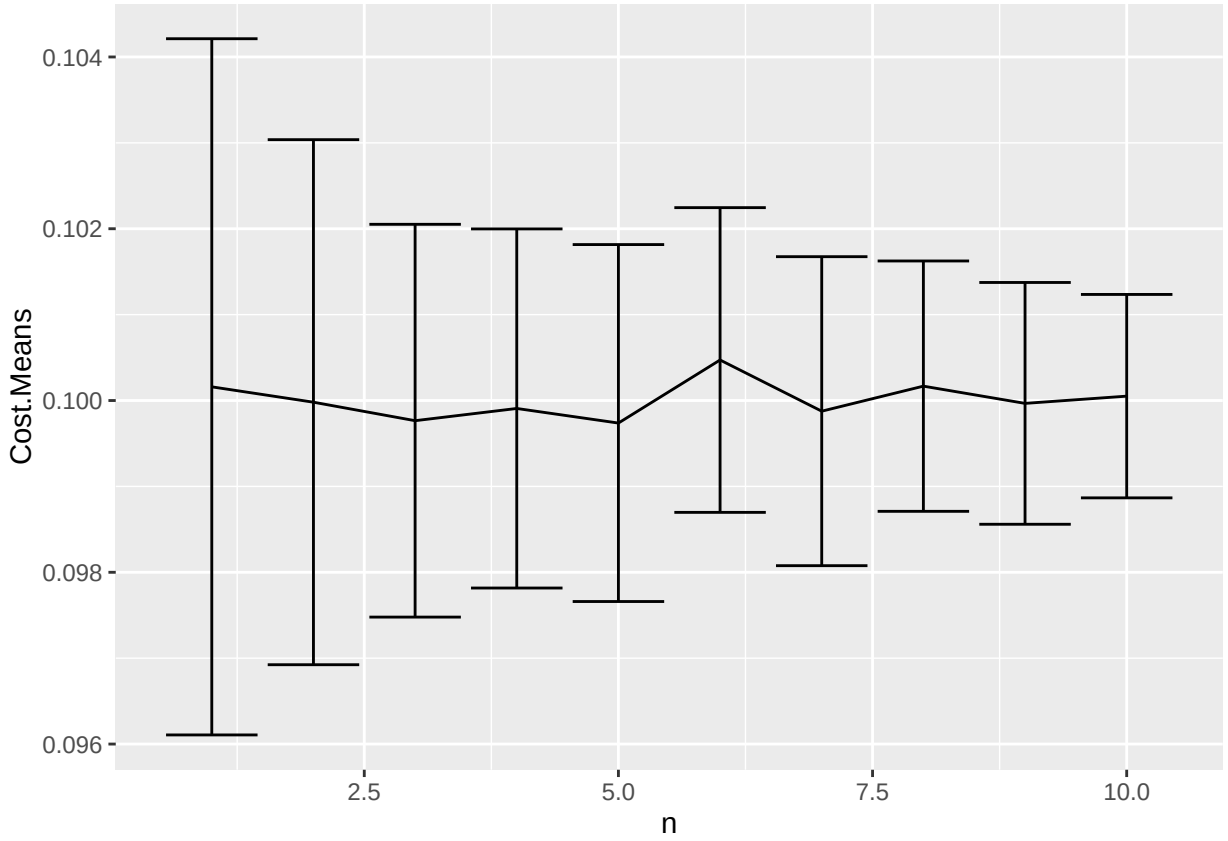
This appears to work (the mean is always close to 1/10). Now to move on to the crude Monte Carlo simulation of the cost function for n sizes from 1000 to 10,000 by increments of 1000.

```
n.increments <- seq(from = 1000, to = 10000, by = 1000)

generate.x.list <- function(n, D){
  x <- as.matrix(runif(D*n, min = -5, max = 5), nrow=n, ncol=D)
  return(x)
}

x.list <- lapply(n.increments, FUN = generate.x.list, D=D)
```

```
crude.cost.sim.1 <- replicate(100, sapply(lapply(n.increments, FUN = generate.x.list, D=D), FUN = cost.
```

```
crude.cost.sim.1.means <- apply(crude.cost.sim.1, 1, FUN = mean)
crude.cost.sim.1.sd <- apply(crude.cost.sim.1, 1, FUN = sd)
n <- 1:10
crude.cost.sim.1.results <- cbind(n, crude.cost.sim.1.means, crude.cost.sim.1.sd) %>% as.data.frame()
colnames(crude.cost.sim.1.results) <- c("n", "Cost.Means", "Cost.Standard.Deviation")
```

```
limits <- aes(ymax = Cost.Means + Cost.Standard.Deviation, ymin = Cost.Means - Cost.Standard.Deviation)
ggplot(crude.cost.sim.1.results, aes(x=n, y=Cost.Means)) + geom_line() + geom_errorbar(limits)
```



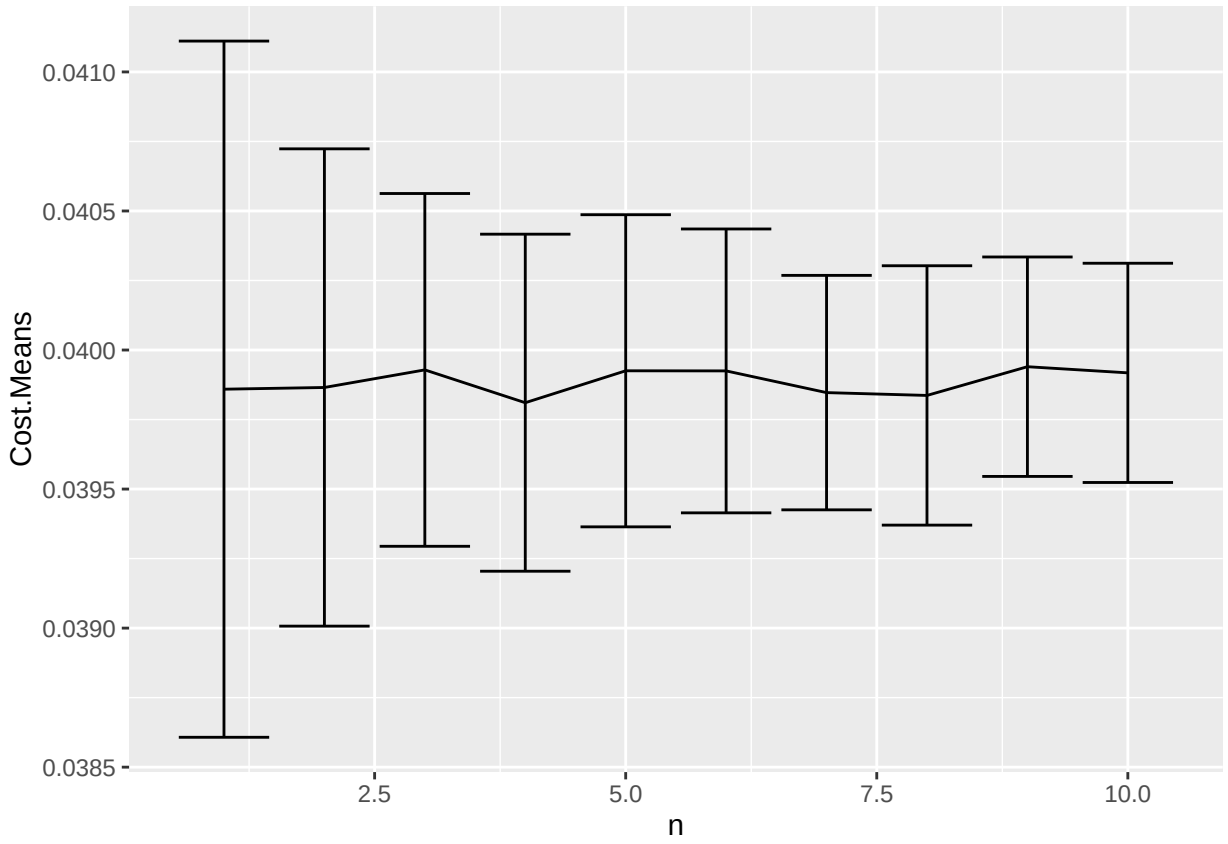
Now for D = 2

```
D.2 = 2
crude.cost.sim.2 <- replicate(100, sapply(lapply(n.increments, FUN = generate.x.list, D = D.2), FUN = co

(target.D.2 <- (1/10)^2)
```

```
## [1] 0.01
```

```
crude.cost.sim.2.means <- apply(crude.cost.sim.2, 1, FUN = mean)
crude.cost.sim.2.sd <- apply(crude.cost.sim.2, 1, FUN = sd)
n <- 1:10
crude.cost.sim.2.results <- cbind(n, crude.cost.sim.2.means, crude.cost.sim.2.sd) %>% as.data.frame()
colnames(crude.cost.sim.2.results) <- c("n", "Cost.Means", "Cost.Standard.Deviation")
```

```
limits.2 <- aes(ymax = Cost.Means + Cost.Standard.Deviation, ymin = Cost.Means - Cost.Standard.Deviation
ggplot(crude.cost.sim.2.results, aes(x=n, y=Cost.Means)) + geom_line() + geom_errorbar(limits.2)
```


0.0410
0.0405
0.0400
0.0395
0.0390
0.0385
Cost.Means
2.5
5.0
7.5
10.0
n