DATA 604 HW 4

Dan Fanelli

10/23/2016

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.2.4
```

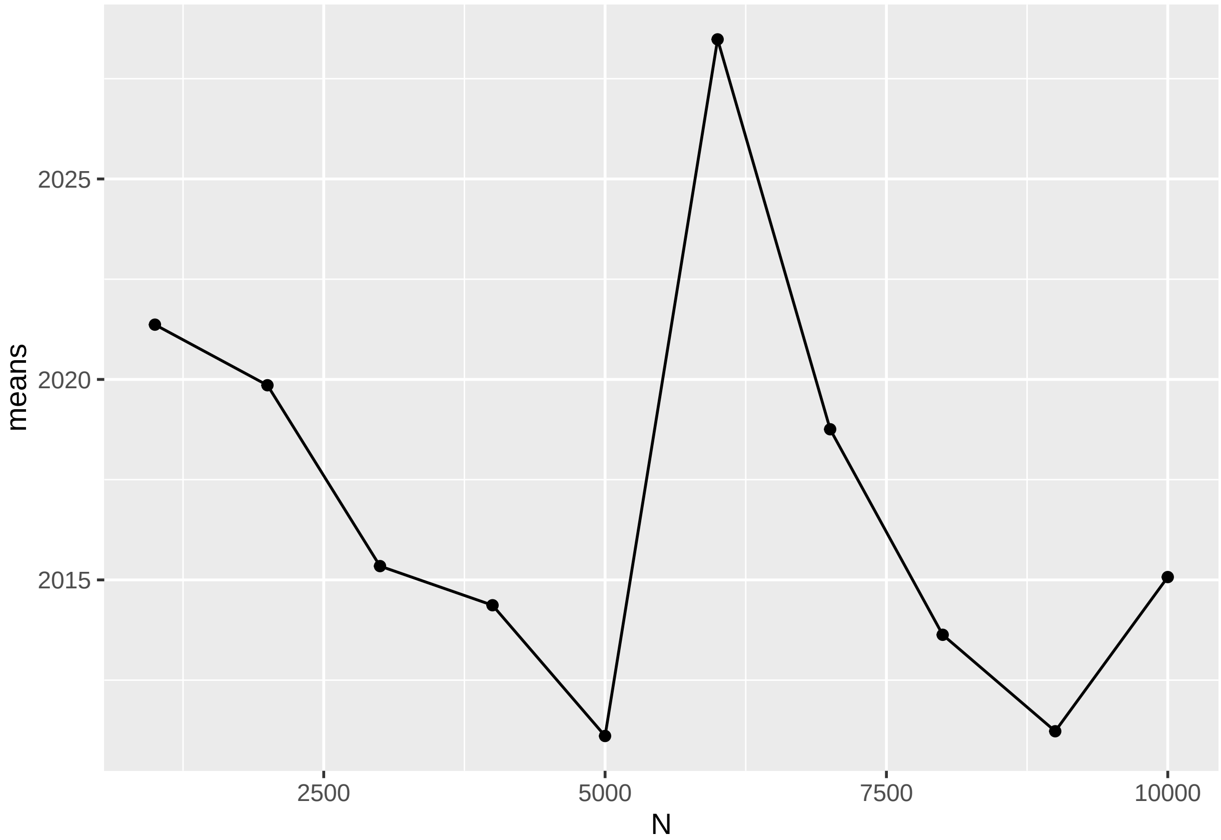
```
library(knitr)

# x is a D-dimensional random variable (i.e., a Dx1 column vector) and each component of x is uniformly
cost_func <- function(x,D=1){
  denominator <- ((2*pi)^(D/2))
  return ((1/denominator)*(exp(1)^(-0.5)*(t(x) %*% x)))
}

truExp <- function(D=1){
  return ((1/10)^D)
}

truExp()
```

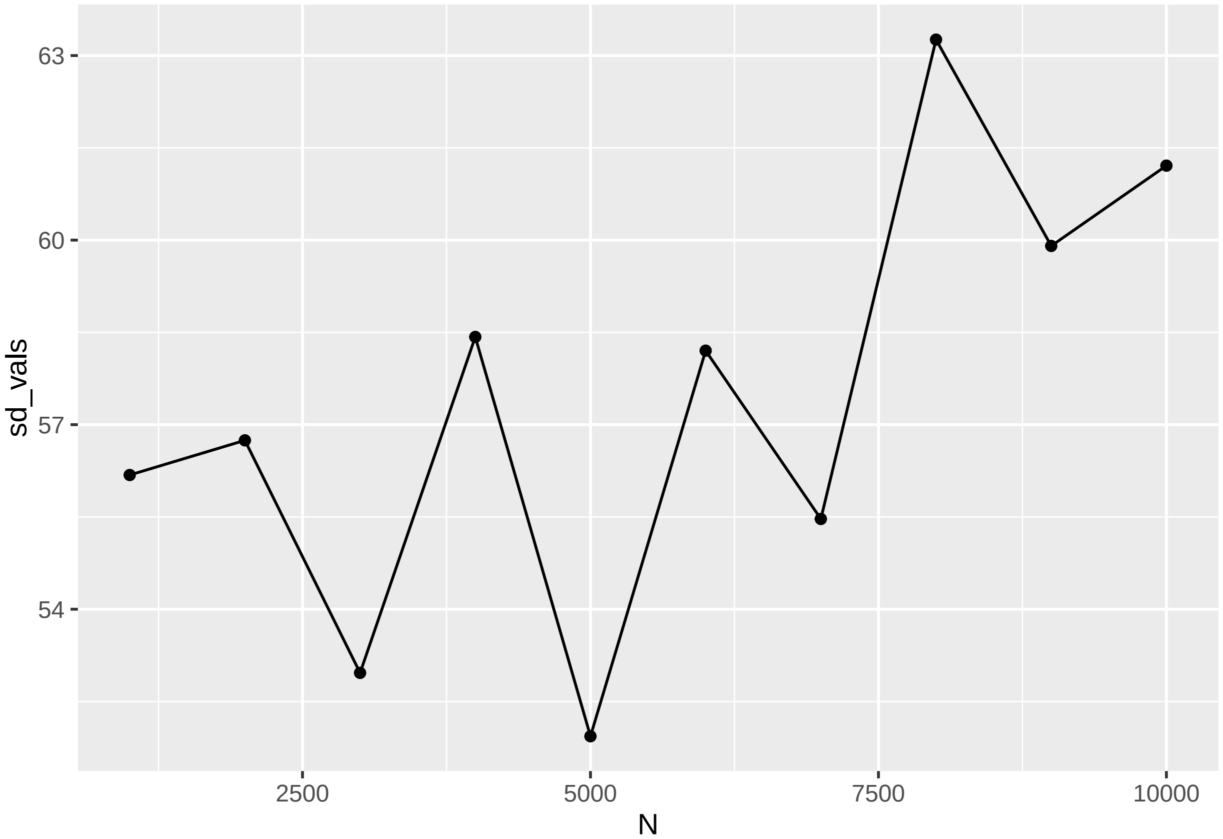
```
## [1] 0.1
```

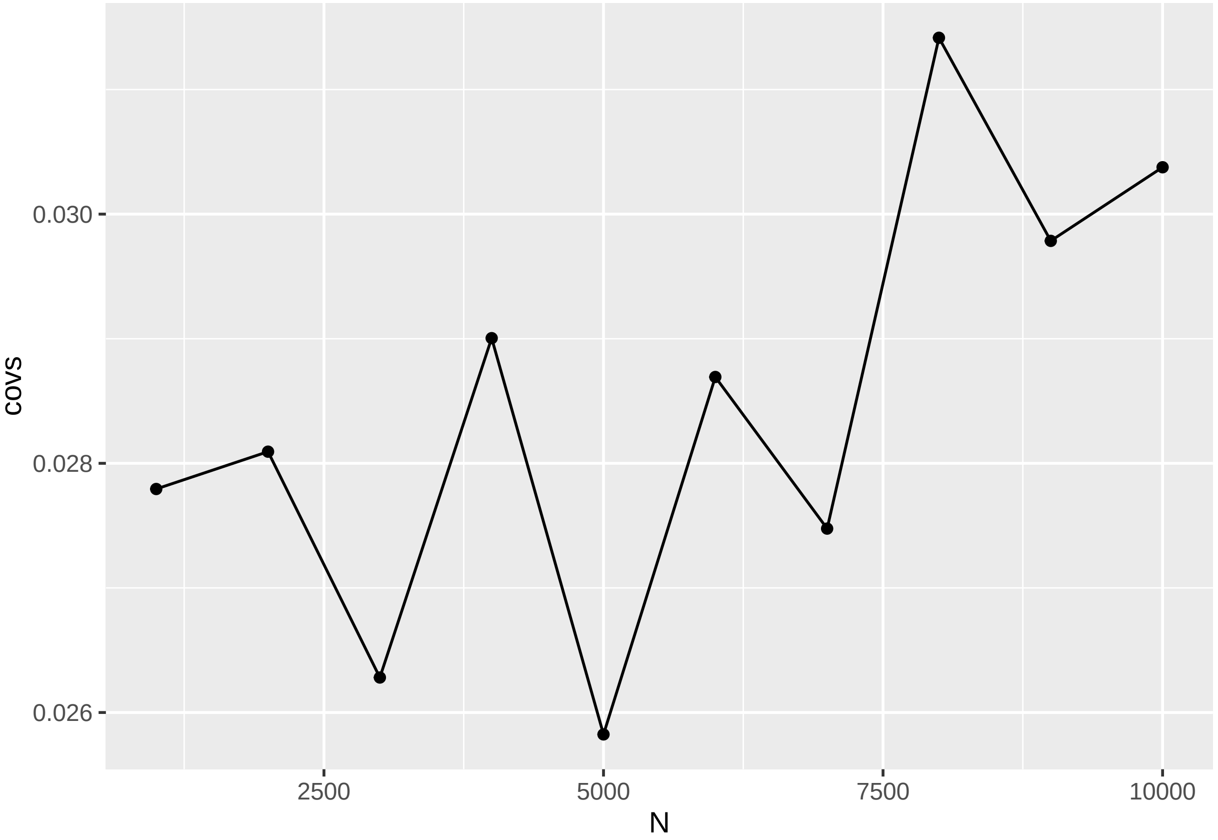
1a - Crude Monte Carlo

```
go_1a <- function(D=1){
  N <- 1000 * c(1:10)
  means <- c()
  sd_vals <- c()
  covs <- c()

  for(n in seq(1000,10000,1000)){
    estimates <- c()
    for(k in 1:100){
      estimates <- c(estimates, cost_func(runif(1000, min = -5, max = 5)))
    }
    #print(estimates)
    means <- c(means, mean(estimates))
    sd_vals <- c(sd_vals, sd(estimates))
    covs <- c(covs, (sd(estimates) / mean(estimates)))
  }
  results <- data.frame(N,means,sd_vals,covs)
  print(results)

  print(ggplot(data=results, aes(x=N, y=means)) + geom_line() + geom_point())
  print(ggplot(data=results, aes(x=N, y=sd_vals)) + geom_line() + geom_point())
  print(ggplot(data=results, aes(x=N, y=covs)) + geom_line() + geom_point())
}

go_1a()
```

```
##        N    means  sd_vals       covs
## 1   1000 2021.367 56.18168 0.02779391
## 2   2000 2019.855 56.74489 0.02809355
## 3   3000 2015.345 52.96503 0.02628088
## 4   4000 2014.368 58.42590 0.02900457
## 5   5000 2011.108 51.93551 0.02582433
## 6   6000 2028.481 58.20300 0.02869290
## 7   7000 2018.757 55.46734 0.02747599
## 8   8000 2013.631 63.25906 0.03141541
## 9   9000 2011.226 59.90533 0.02978548
## 10 10000 2015.071 61.21023 0.03037621
```




63
60
57
54
sd_vals
2500
5000
7500
10000
N

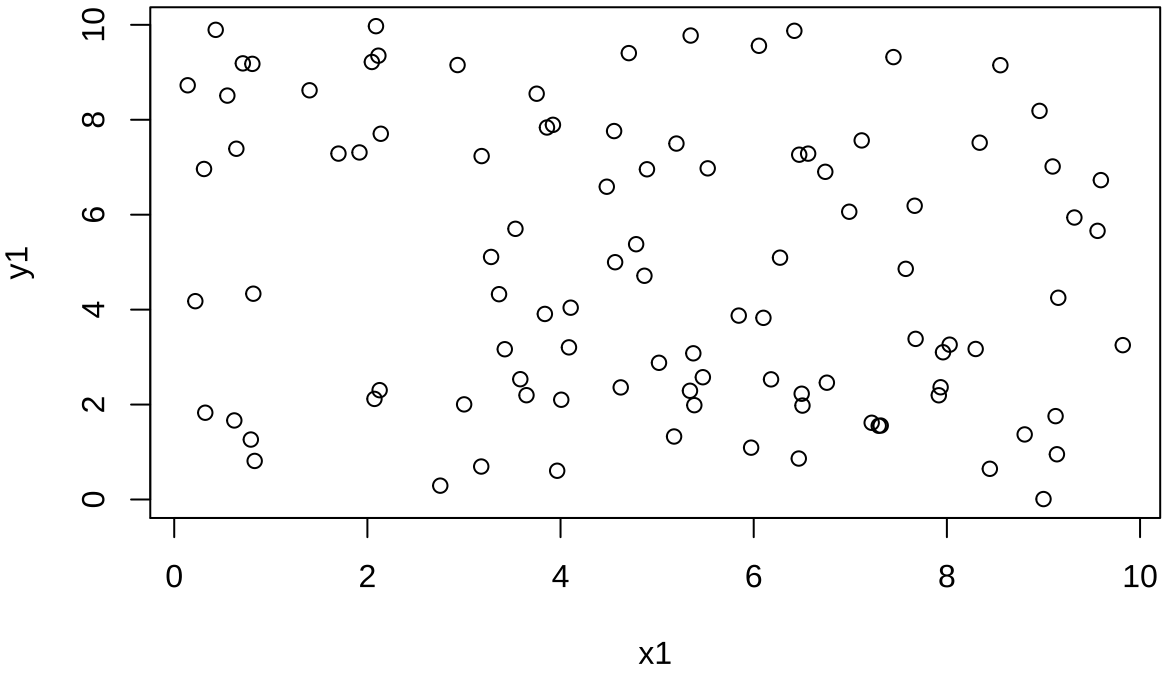

0.030
0.028
0.026
covs
2500
5000
7500
10000
N

1b - Quasi Random Numbers

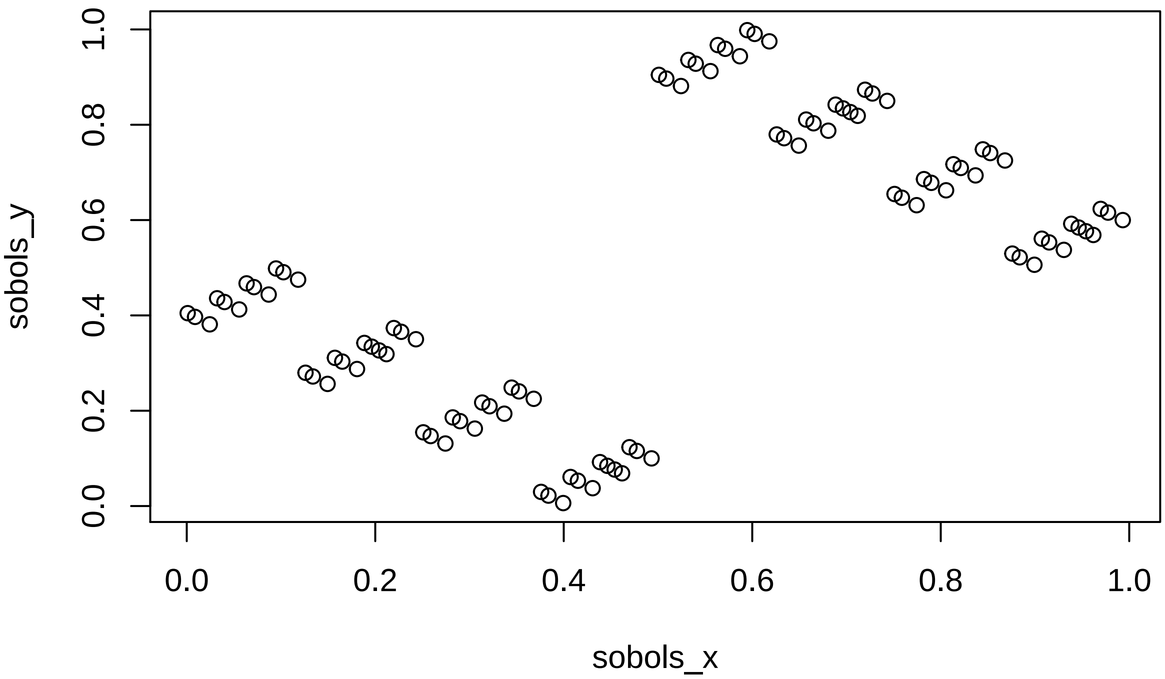
```
library(qrng)
```

```
## Warning: package 'qrng' was built under R version 3.2.5
```

```
x1 <- runif(100,0,10)
y1 <- runif(100,0,10)

plot(x1, y1)
```



```
sobols_x <- sobol(n=100, d=1, randomize = TRUE)
sobols_y <- sobol(n=100, d=1, randomize = TRUE)

plot(sobols_x, sobols_y)
```

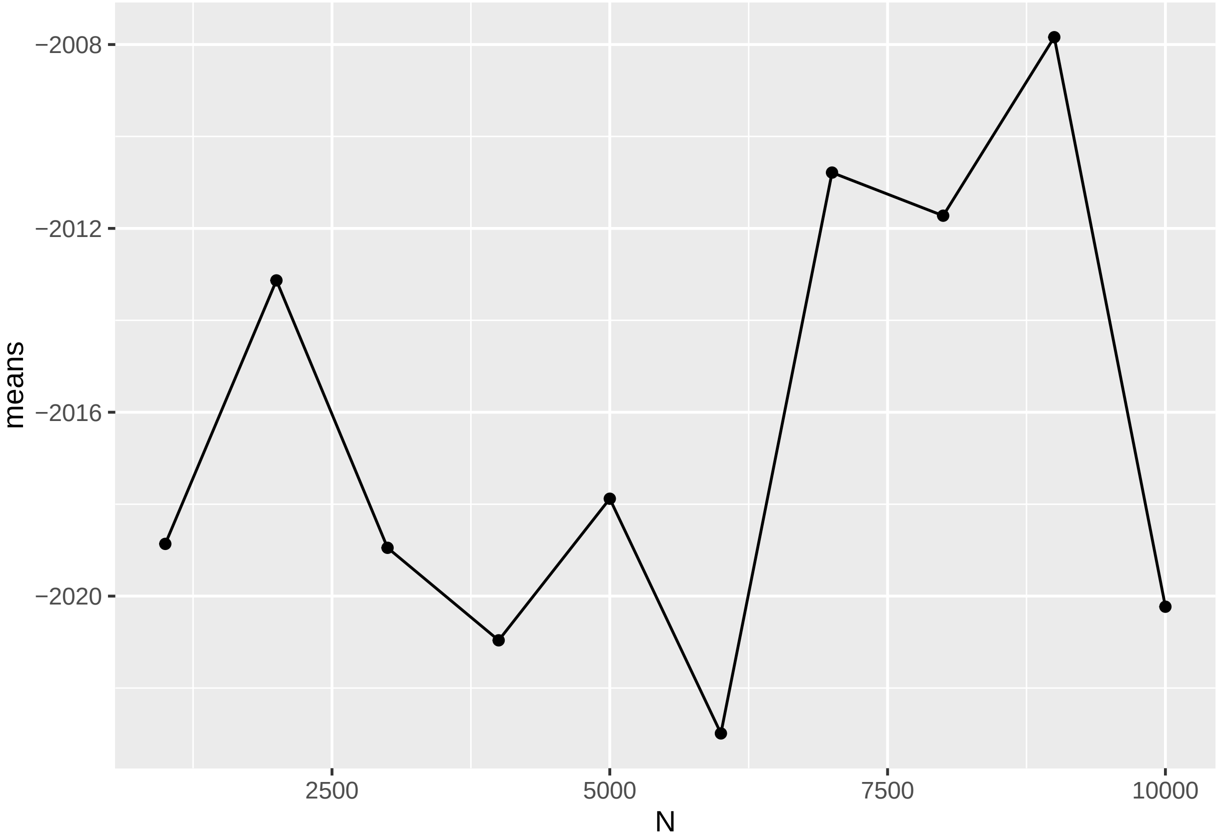


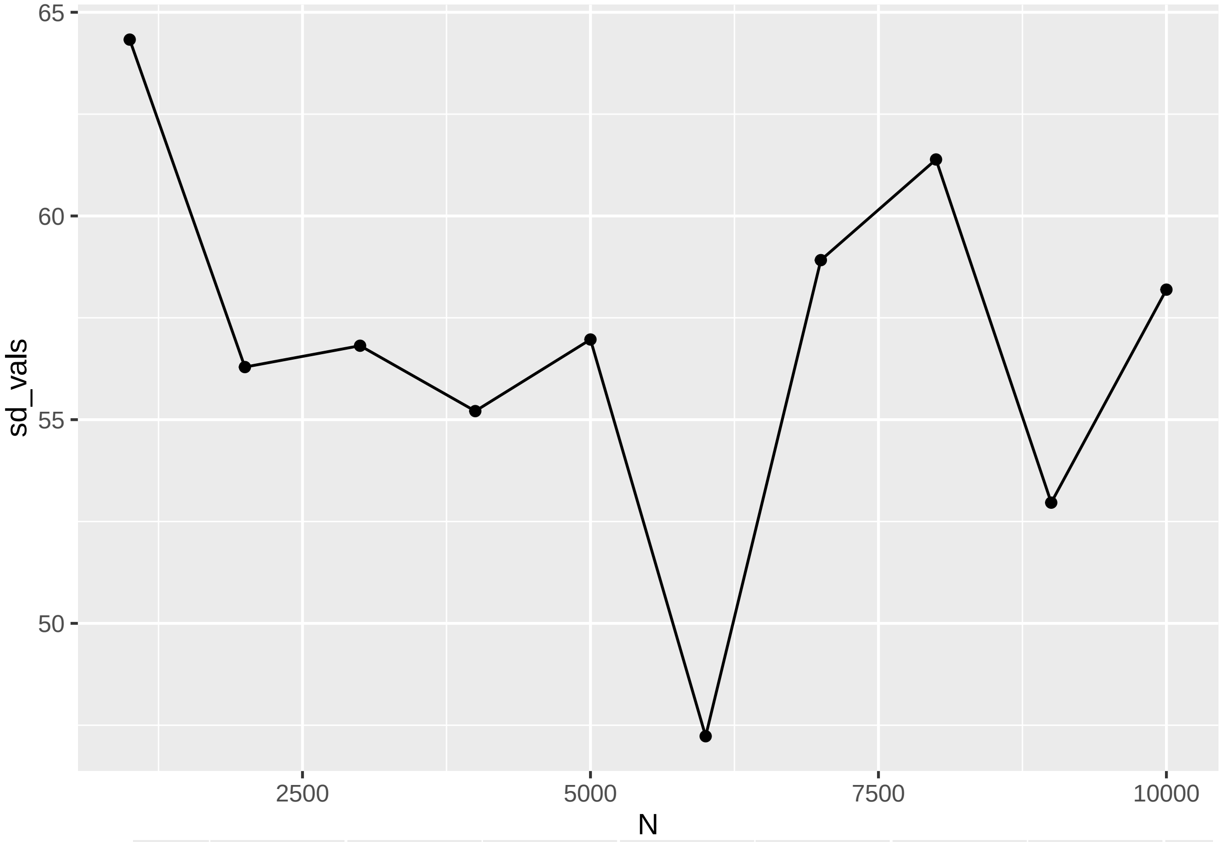
1c - Antithetic Variates

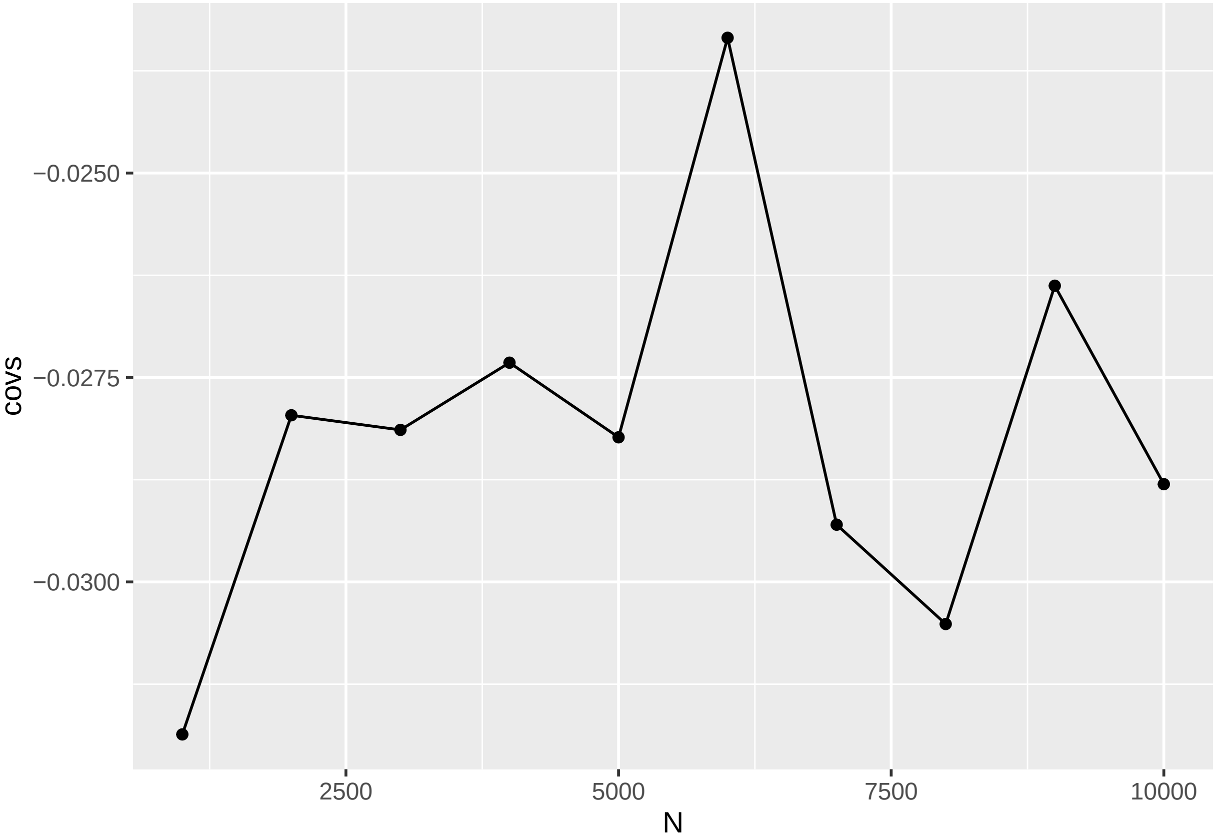
```
go_1c <- function(D=1){
  N <- 1000 * c(1:10)
  means <- c()
  sd_vals <- c()
  covs <- c()

  for(n in seq(1000,10000,1000)){
    estimates <- c()
    for(k in 1:100){
      positives <- cost_func(runif(1000, min = -5, max = 5))
      negatives <- positives * (-1)
      #estimates <- c(estimates, positives, positives * (-1))
      estimates <- c(estimates, negatives)
    }
    #print(estimates)
    means <- c(means, mean(estimates))
    sd_vals <- c(sd_vals, sd(estimates))
    covs <- c(covs, (sd(estimates) / mean(estimates)))
  }
  results <- data.frame(N,means,sd_vals,covs)
  print(results)

  print(ggplot(data=results, aes(x=N, y=means)) + geom_line() + geom_point())
  print(ggplot(data=results, aes(x=N, y=sd_vals)) + geom_line() + geom_point())
  print(ggplot(data=results, aes(x=N, y=covs)) + geom_line() + geom_point())
}
```

```
go_1c()
```

```
##        N     means  sd_vals        covs
## 1   1000 -2018.863 64.32897 -0.03186395
## 2   2000 -2013.132 56.29086 -0.02796184
## 3   3000 -2018.949 56.81487 -0.02814081
## 4   4000 -2020.962 55.21023 -0.02731879
## 5   5000 -2017.881 56.96759 -0.02823140
## 6   6000 -2022.987 47.22925 -0.02334630
## 7   7000 -2010.786 58.91629 -0.02930013
## 8   8000 -2011.724 61.38651 -0.03051439
## 9   9000 -2007.841 52.96255 -0.02637786
## 10 10000 -2020.231 58.19164 -0.02880445
```




65
60
55
50
sd_vals
2500
5000
7500
10000
N


−0.0250
−0.0275
−0.0300
covs
2500
5000
7500
10000
N

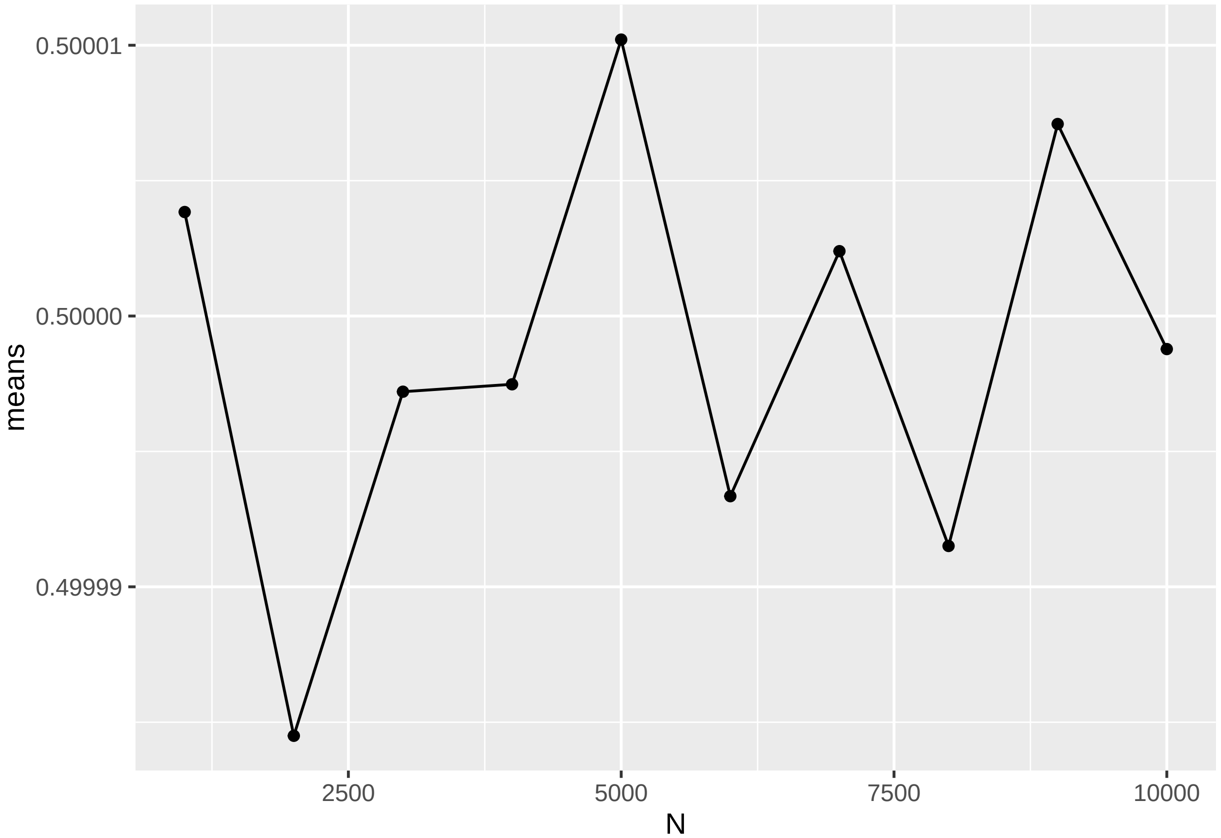
1d - Latin Hypercube Sampling

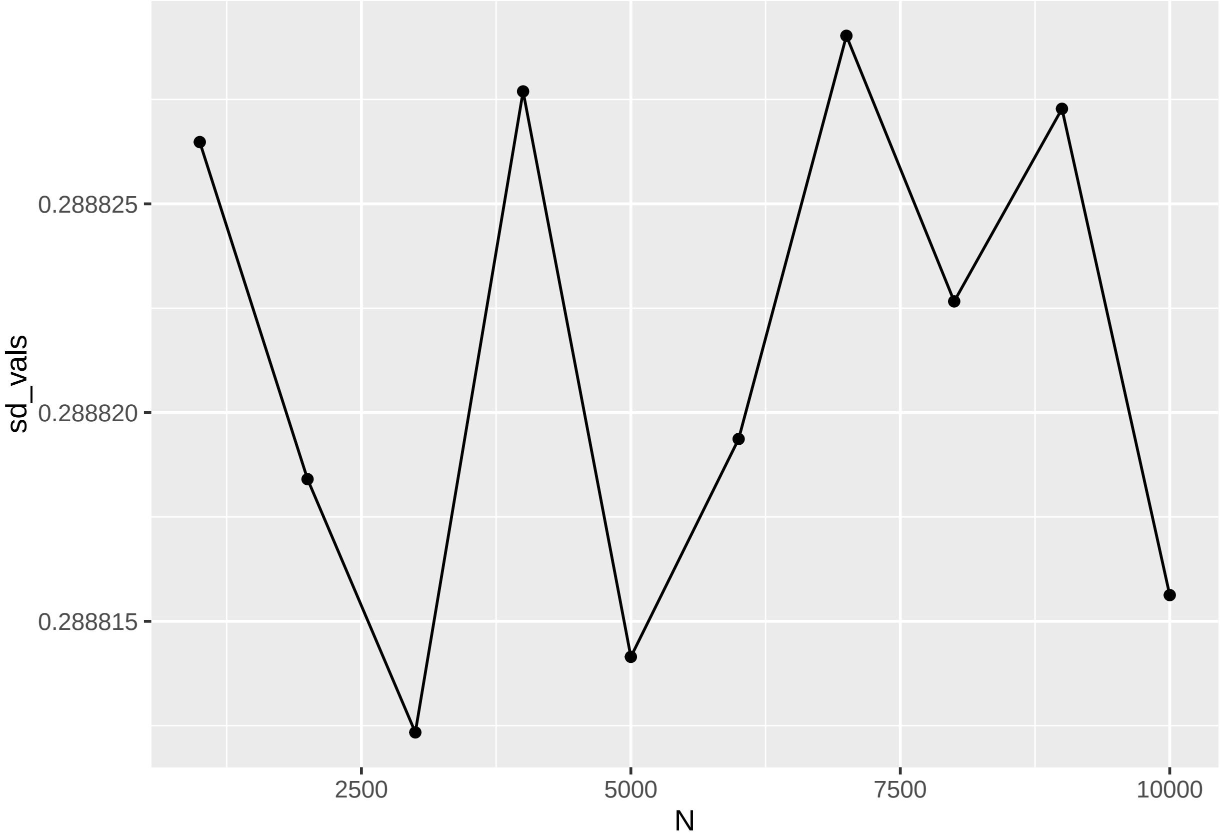
```
library(lhs)
```

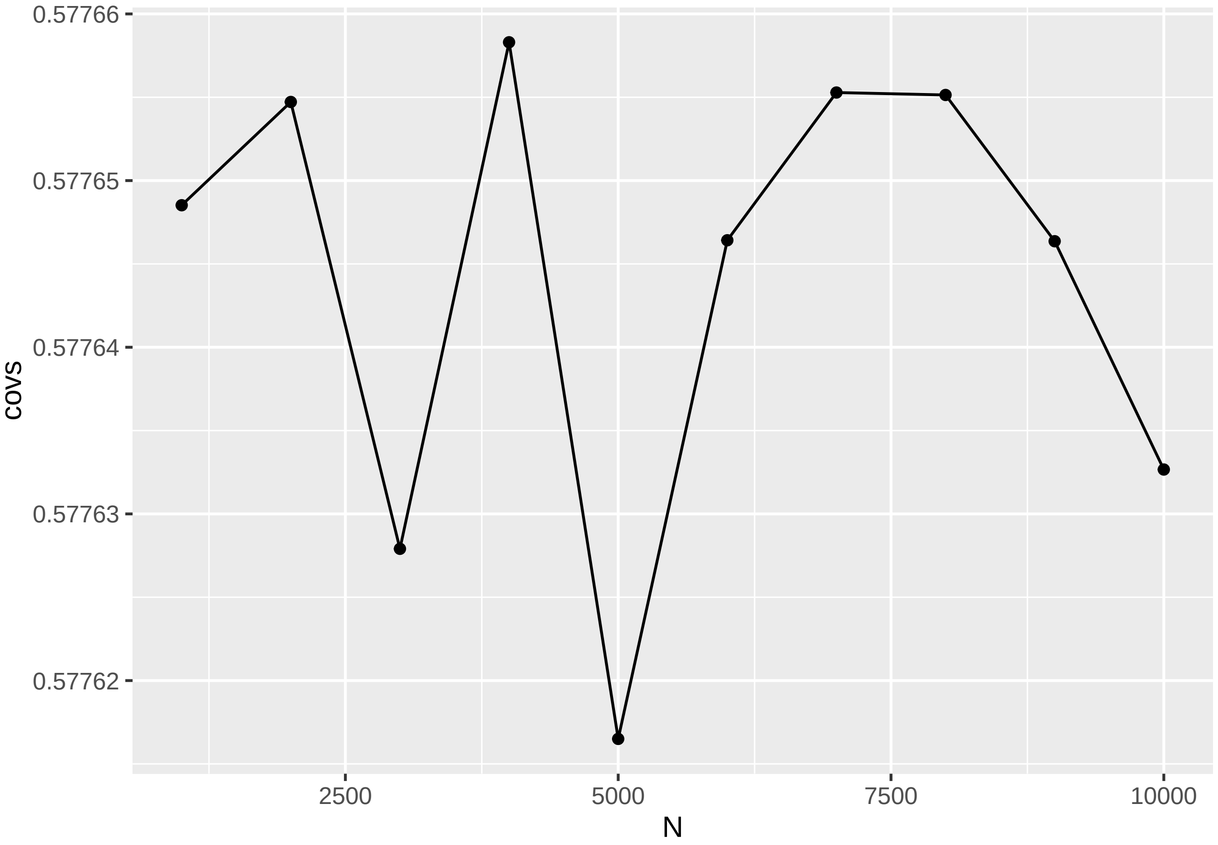
```
## Warning: package 'lhs' was built under R version 3.2.5
```

```
go_1d <- function(D=1){
  N <- 1000 * c(1:10)
  means <- c()
  sd_vals <- c()
  covs <- c()

  for(n in seq(1000,10000,1000)){
    estimates <- c()
    for(k in 1:100){
      estimates <- randomLHS(1000, 1)
      #estimates <- c(estimates, cost_func(runif(1000, min = -5, max = 5)))
    }
    #print(estimates)
    means <- c(means, mean(estimates))
    sd_vals <- c(sd_vals, sd(estimates))
    covs <- c(covs, (sd(estimates) / mean(estimates)))
  }
  results <- data.frame(N,means,sd_vals,covs)
  print(results)

  print(ggplot(data=results, aes(x=N, y=means)) + geom_line() + geom_point())
  print(ggplot(data=results, aes(x=N, y=sd_vals)) + geom_line() + geom_point())
  print(ggplot(data=results, aes(x=N, y=covs)) + geom_line() + geom_point())
}

go_1d()
```

```
##        N     means   sd_vals      covs
## 1   1000 0.5000038 0.2888265 0.5776485
## 2   2000 0.4999845 0.2888184 0.5776547
## 3   3000 0.4999972 0.2888123 0.5776279
## 4   4000 0.4999975 0.2888277 0.5776583
## 5   5000 0.5000102 0.2888141 0.5776165
## 6   6000 0.4999933 0.2888194 0.5776464
## 7   7000 0.5000024 0.2888290 0.5776553
## 8   8000 0.4999915 0.2888227 0.5776551
## 9   9000 0.5000071 0.2888273 0.5776464
## 10 10000 0.4999988 0.2888156 0.5776327
```


0.50001
0.50000
0.49999
means
2500
5000
7500
10000
N


0.288825
0.288820
0.288815
sd_vals
2500
5000
7500
10000
N



1e - Importance Sampling

1f - Summary

All Code:

```
library(ggplot2)
library(knitr)

# x is a D-dimensional random variable (i.e., a Dx1 column vector) and each component of x is uniformly
cost_func <- function(x,D=1){
  denominator <- ((2*pi)^(D/2))
  return ((1/denominator)*(exp(1)^(-0.5)*(t(x) %*% x)))
}

truExp <- function(D=1){
  return ((1/10)^D)
}

truExp()
go_1a <- function(D=1){
  N <- 1000 * c(1:10)
  means <- c()
  sd_vals <- c()
  covs <- c()

  for(n in seq(1000,10000,1000)){
    estimates <- c()
    for(k in 1:100){
      estimates <- c(estimates, cost_func(runif(1000, min = -5, max = 5)))
    }
    #print(estimates)
    means <- c(means, mean(estimates))
    sd_vals <- c(sd_vals, sd(estimates))
    covs <- c(covs, (sd(estimates) / mean(estimates)))
  }
  results <- data.frame(N,means,sd_vals,covs)
  print(results)

  print(ggplot(data=results, aes(x=N, y=means)) + geom_line() + geom_point())
  print(ggplot(data=results, aes(x=N, y=sd_vals)) + geom_line() + geom_point())
  print(ggplot(data=results, aes(x=N, y=covs)) + geom_line() + geom_point())
}

go_1a()

library(qrng)

x1 <- runif(100,0,10)
y1 <- runif(100,0,10)

plot(x1, y1)

sobols_x <- sobol(n=100, d=1, randomize = TRUE)
sobols_y <- sobol(n=100, d=1, randomize = TRUE)

plot(sobols_x, sobols_y)
go_1c <- function(D=1){
  N <- 1000 * c(1:10)
  means <- c()
  sd_vals <- c()
  covs <- c()

  for(n in seq(1000,10000,1000)){
    estimates <- c()
    for(k in 1:100){
      positives <- cost_func(runif(1000, min = -5, max = 5))
      negatives <- positives * (-1)
      #estimates <- c(estimates, positives, positives * (-1))
      estimates <- c(estimates, negatives)
    }
    #print(estimates)
    means <- c(means, mean(estimates))
    sd_vals <- c(sd_vals, sd(estimates))
    covs <- c(covs, (sd(estimates) / mean(estimates)))
  }
  results <- data.frame(N,means,sd_vals,covs)
  print(results)

  print(ggplot(data=results, aes(x=N, y=means)) + geom_line() + geom_point())
  print(ggplot(data=results, aes(x=N, y=sd_vals)) + geom_line() + geom_point())
  print(ggplot(data=results, aes(x=N, y=covs)) + geom_line() + geom_point())
}

go_1c()

library(lhs)
go_1d <- function(D=1){
  N <- 1000 * c(1:10)
  means <- c()
  sd_vals <- c()
  covs <- c()

  for(n in seq(1000,10000,1000)){
    estimates <- c()
    for(k in 1:100){
      estimates <- randomLHS(1000, 1)
      #estimates <- c(estimates, cost_func(runif(1000, min = -5, max = 5)))
    }
    #print(estimates)
    means <- c(means, mean(estimates))
    sd_vals <- c(sd_vals, sd(estimates))
    covs <- c(covs, (sd(estimates) / mean(estimates)))
  }
  results <- data.frame(N,means,sd_vals,covs)
  print(results)

  print(ggplot(data=results, aes(x=N, y=means)) + geom_line() + geom_point())
  print(ggplot(data=results, aes(x=N, y=sd_vals)) + geom_line() + geom_point())
  print(ggplot(data=results, aes(x=N, y=covs)) + geom_line() + geom_point())
}

go_1d()
##
```