IS 604 Assignment 4

David Stern

October 15, 2015

1. In this problem, we will implement and investigate a series of variance reduction procedures for Monte Carlo methods by estimating the expected value of a cost function $c(x)$ which depends on a D-dimensional random variable x.

$$c(x) = \frac{1}{(2\pi)^{\frac{D}{2}}} e^{-\frac{1}{2}x^T x}$$

$$x_i \; U(-5,5) \; for \; i = 1..D$$

$$Analytically, \; E[c(x)] = (\frac{1}{10})^D$$

That is, x is a D-dimensional random variable (i.e., a $D \times 1$ column vector) and each component of x is uniformly distributed between -5 and 5.

a) Crude Monte Carlo

For sample sizes of 1000 to 10000 (in increments of 1000), obtain 100 estimates for $E[c(x)]$ when $D = 1$, using crude Monte Carlo sampling. Calculate the average value of the 100 estimates as well as their standard deviation, and plot them. In your plot, include a line showing the analytical value for $E[c(x)]$.

This function will give us the expected value of the cost function with inputs D for the dimension of the random variable and N for the sample size.

```
EX <- function(D,N){
  c <- numeric(N)
  for (i in 1:N){
    x <- runif(D,-5,5) # create D-dimensional random variable with elements from uniform distribution
    c[i] <- exp(-0.5*sum(t(x)*x))/((2*pi)^(D/2)) # cost function
  }
  mean(c)
}
```

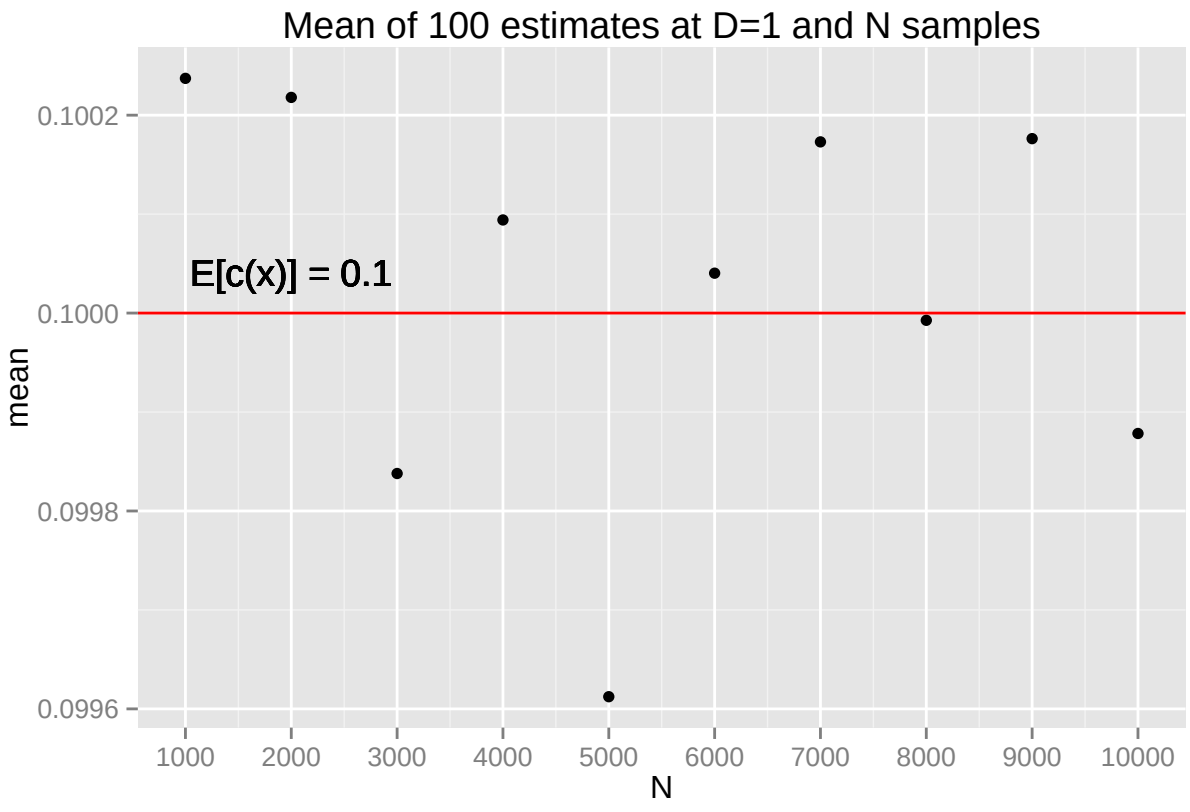
Now we will create a matrix with a column for each sample size N and 100 rows to store estimates from each iteration of the function. We will then apply column-wise functions to find the mean and standard deviation of the 100 estimates for each N. Here we see that even at 1000 samples, we get a very close approximation to the analytical value of 0.1 for $E[c(x)]$ at $D = 1$.

```
suppressWarnings(suppressMessages(library(knitr)))
N <- seq(1000,10000,by=1000)
estimatesD1 <- matrix(data=NA, nrow=100, ncol=length(N))
colnames(estimatesD1) <- N
rownames(estimatesD1) <- 1:100
for(i in 1:length(N)){
  for (j in 1:100){
    estimatesD1[j,i] <- EX(1,N[i]) # D = 1
  }
```

```
}
meanD1 <- apply(estimatesD1,2,mean) # mean of estimates matrix, column-wise by sample size
stdD1 <- apply(estimatesD1,2,sd)
cvD1 <- meanD1/stdD1
D1 <- t(data.frame("mean"=meanD1,"sd"=stdD1,"cv"=cvD1))
summary <- data.frame() #create dataframe for part f summary
summary <- rbind(summary,D1[,1])
summary <- rbind(summary,D1[,4])
kable(D1)
```

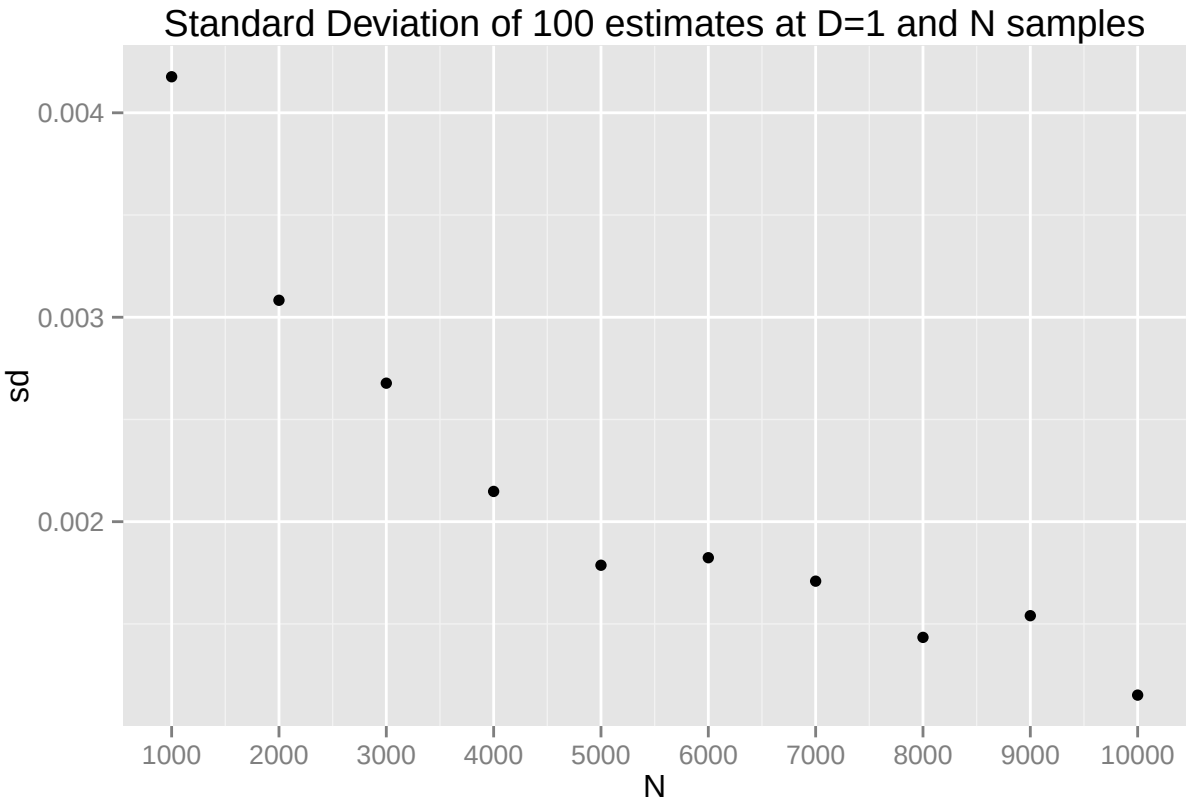
	1000	2000	3000	4000	5000	6000	7000	8000	9
mean	0.1002372	0.100218	0.0998379	0.1000941	0.0996123	0.1000403	0.1001730	0.0999927	0.1001
sd	0.0041763	0.003083	0.0026775	0.0021480	0.0017870	0.0018238	0.0017092	0.0014342	0.0015
cv	24.0016141	32.506461	37.2877106	46.5978000	55.7434771	54.8528731	58.6096055	69.7193584	65.0407

```
suppressWarnings(suppressMessages(library(ggplot2)))
D1 <- as.data.frame(t(D1))
ggplot(D1) + geom_point(aes(x=N,y=mean,label="mean")) +
  geom_hline(yintercept=0.1,color="red") +
  geom_text(aes(2000,0.1,label = "E[c(x)] = 0.1", vjust = -1)) +
  scale_x_continuous(breaks=N) +
  ggtitle("Mean of 100 estimates at D=1 and N samples")
```




```
ggplot(D1) + geom_point(aes(x=N,y=sd,label="standard deviation")) +
  scale_x_continuous(breaks=N) +
```

```
  ggtitle("Standard Deviation of 100 estimates at D=1 and N samples")
```

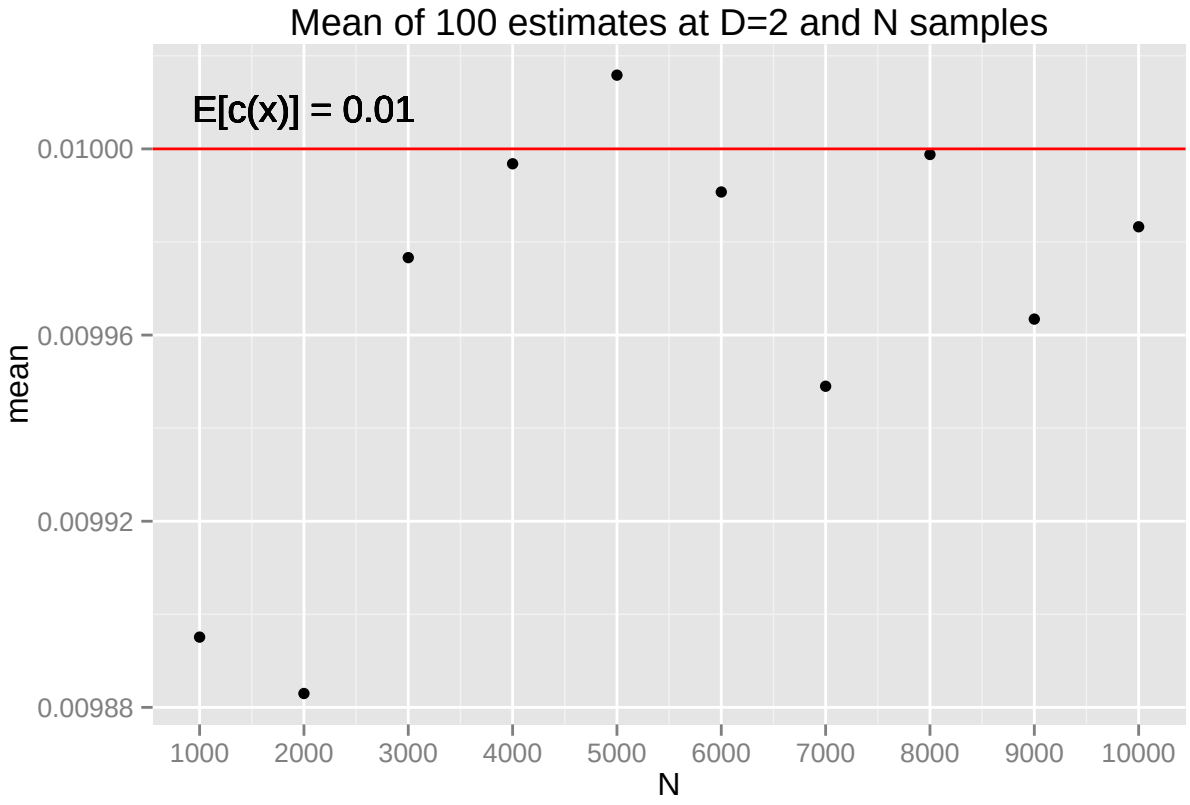


Now we will repeat this analysis for $D = 2$.

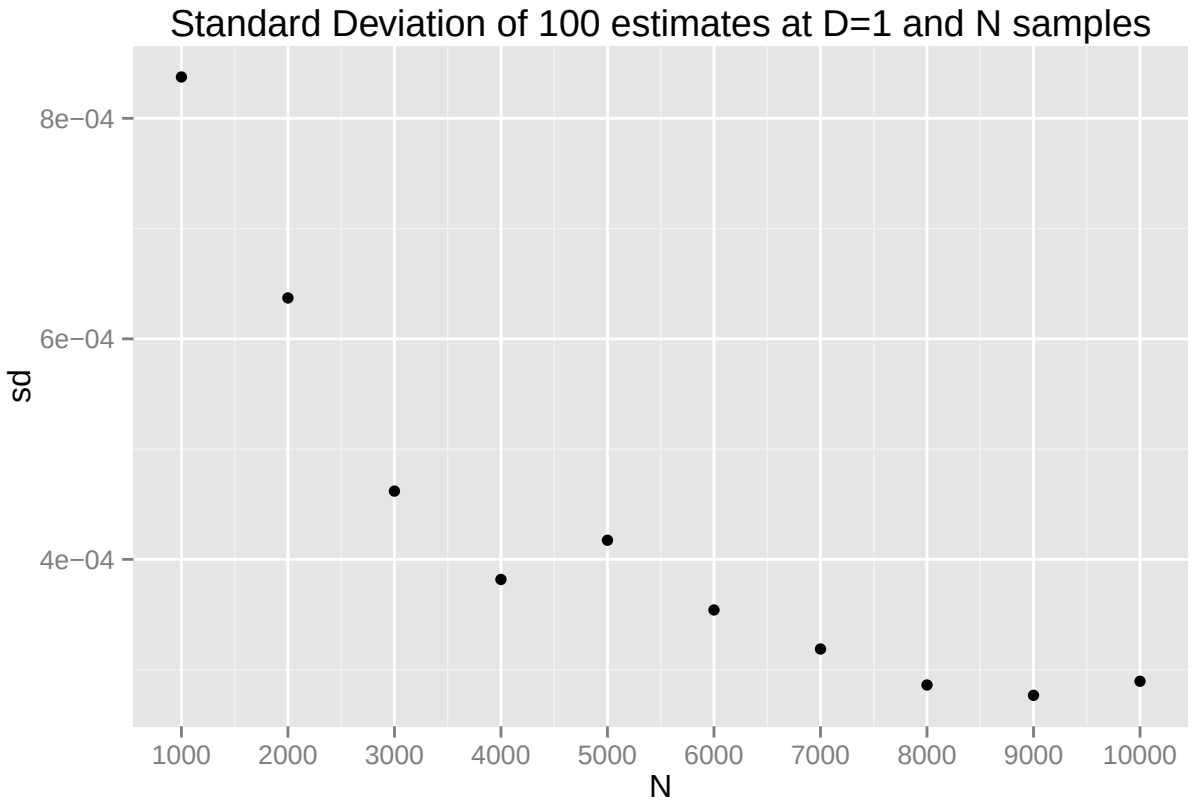
```
estimatesD2 <- matrix(data=NA, nrow=100, ncol=length(N))
colnames(estimatesD2) <- N
rownames(estimatesD2) <- 1:100
for(i in 1:length(N)){
  for (j in 1:100){
    estimatesD2[j,i] <- EX(2,N[i]) # D = 2
  }
}
meanD2 <- apply(estimatesD2,2,mean)
stdD2 <- apply(estimatesD2,2,sd)
cvD2 <- meanD2/stdD2
D2 <- t(data.frame("mean"=meanD2,"sd"=stdD2,"cv"=cvD2))
summary <- rbind(summary,D2[,1])
summary <- rbind(summary,D2[,4])
kable(D2)
```

	1000	2000	3000	4000	5000	6000	7000	8000	
mean	0.0098951	0.0098830	0.0099766	0.0099968	0.0100158	0.0099907	0.0099490	0.0099988	0.009
sd	0.0008375	0.0006371	0.0004619	0.0003818	0.0004173	0.0003541	0.0003186	0.0002860	0.000
cv	11.8150020	15.5126288	21.5995841	26.1853115	24.0013263	28.2148828	31.2224532	34.9562239	36.014

```
D2 <- as.data.frame(t(D2))
ggplot(D2) + geom_point(aes(x=N,y=mean,label="mean")) +
  geom_hline(yintercept=0.01,color="red") +
  geom_text(aes(2000,0.01,label = "E[c(x)] = 0.01", vjust = -1)) +
  scale_x_continuous(breaks=N) +
  ggtitle("Mean of 100 estimates at D=2 and N samples")
```




```
ggplot(D2) + geom_point(aes(x=N,y=sd,label="standard deviation")) +
  scale_x_continuous(breaks=N) +
  ggtitle("Standard Deviation of 100 estimates at D=1 and N samples")
```

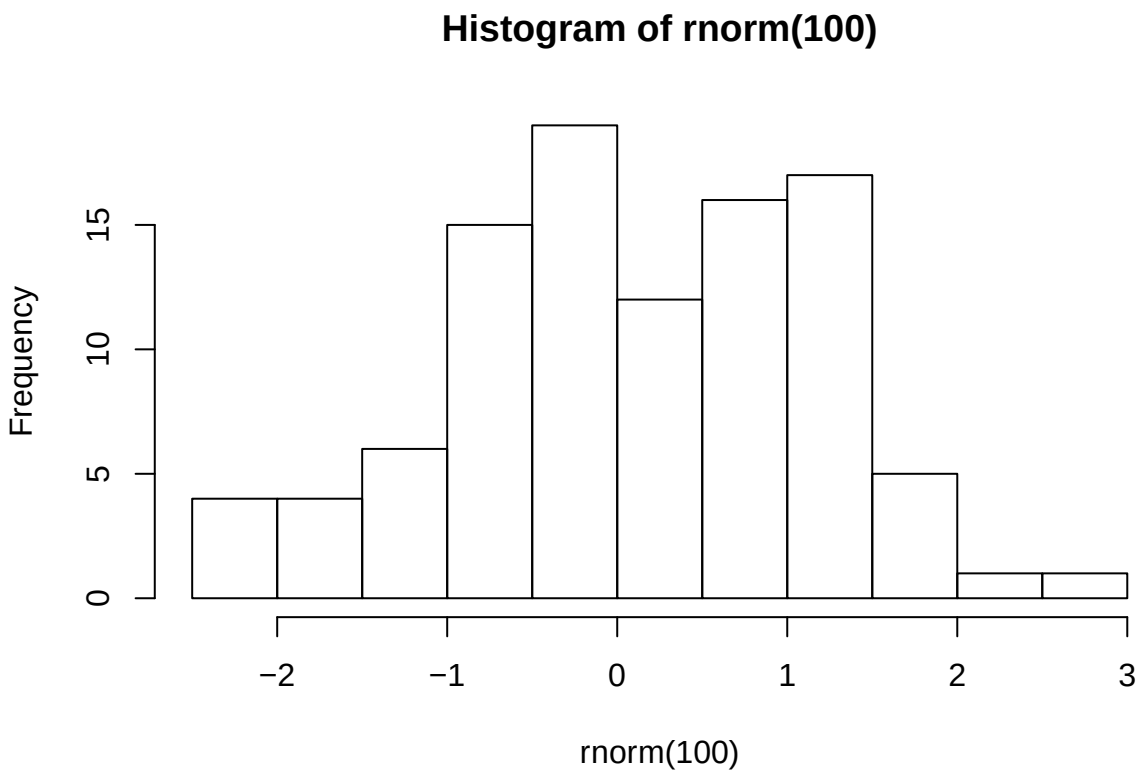


b) Quasi-Random Numbers

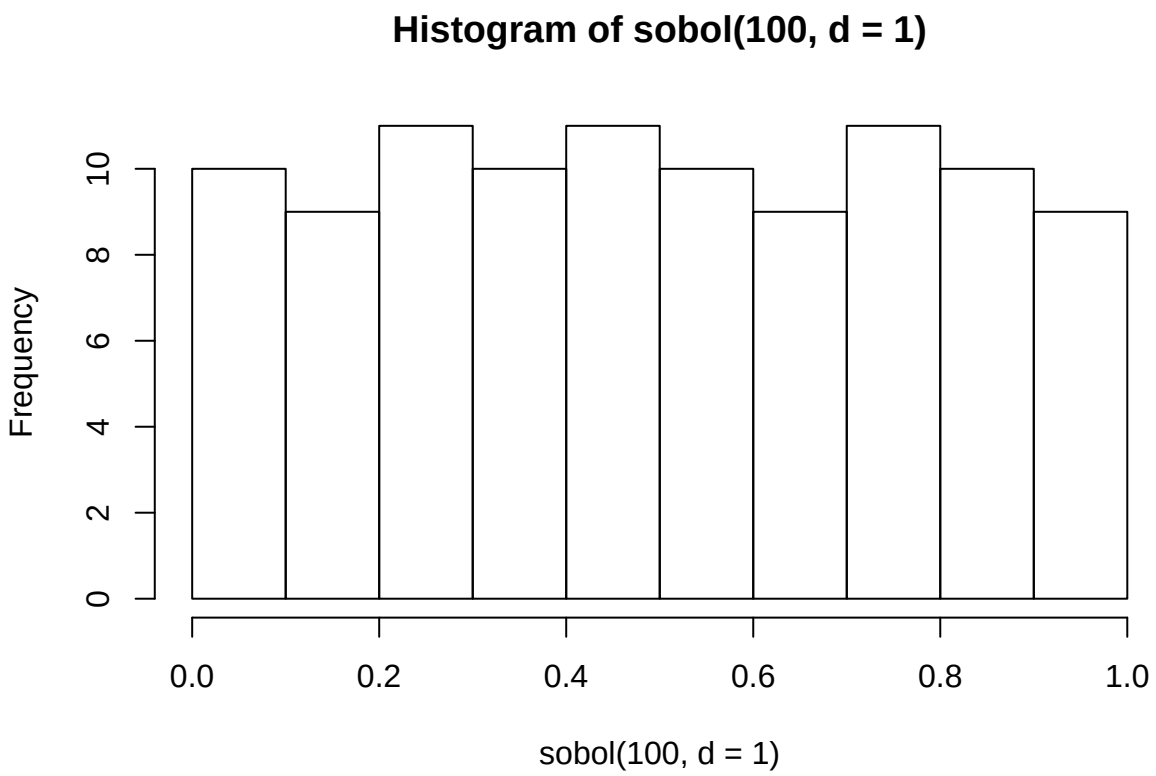
We now investigate the variance reduction properties of quasi-random numbers using Sobol numbers. To see how these Sobol numbers differ from normal random numbers, we will generate and plot 100 pairs of both.

The Sobol QRNG in the R package *randtoolbox* defaults to a uniform a distribution between 0 and 1, with optional scrambling methods. This contrast with the numbers generated from the normal random distribution, which have a mean of 0 and a standard deviation of 1.

```
suppressWarnings(suppressMessages(library(randtoolbox)))
hist(rnorm(100))
```




```
hist(sobol(100,d=1))
```



Repeat the analysis of part a) for $D = 1$ and $D = 2$ using Sobol quasi-random numbers. How does the use of Sobol numbers change the average value and standard deviation of your estimates in $D = 1$ and $D = 2$?

To modify our approach, we first need to recognize that the Sobol distribution is uniform on the interval $(0, 1)$. We will therefore have to generate our random numbers from the Sobol distribution and then scale them over the interval $(-5, 5)$.

```
sobol(10,1)
```

```
##  [1] 0.5000 0.7500 0.2500 0.3750 0.8750 0.6250 0.1250 0.1875 0.6875 0.9375
```

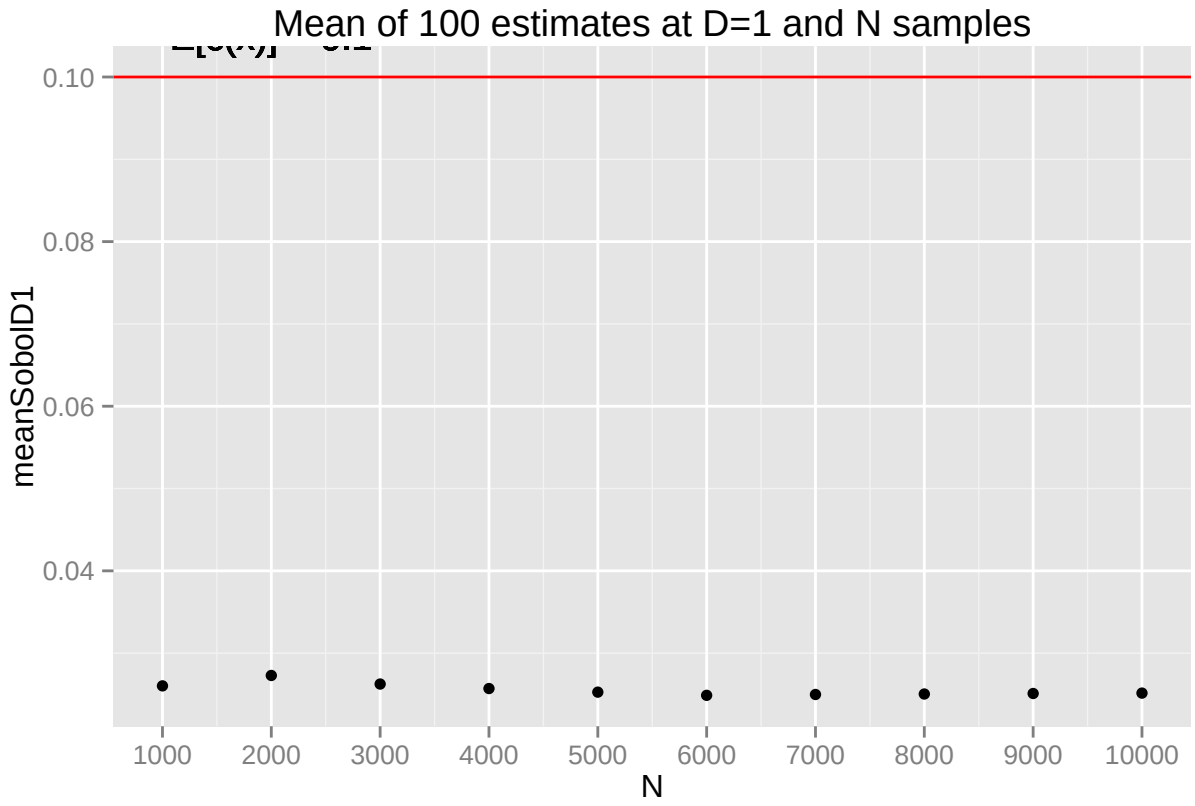
```
sobol(10,1)
```

```
##  [1] 0.5000 0.7500 0.2500 0.3750 0.8750 0.6250 0.1250 0.1875 0.6875 0.9375
```

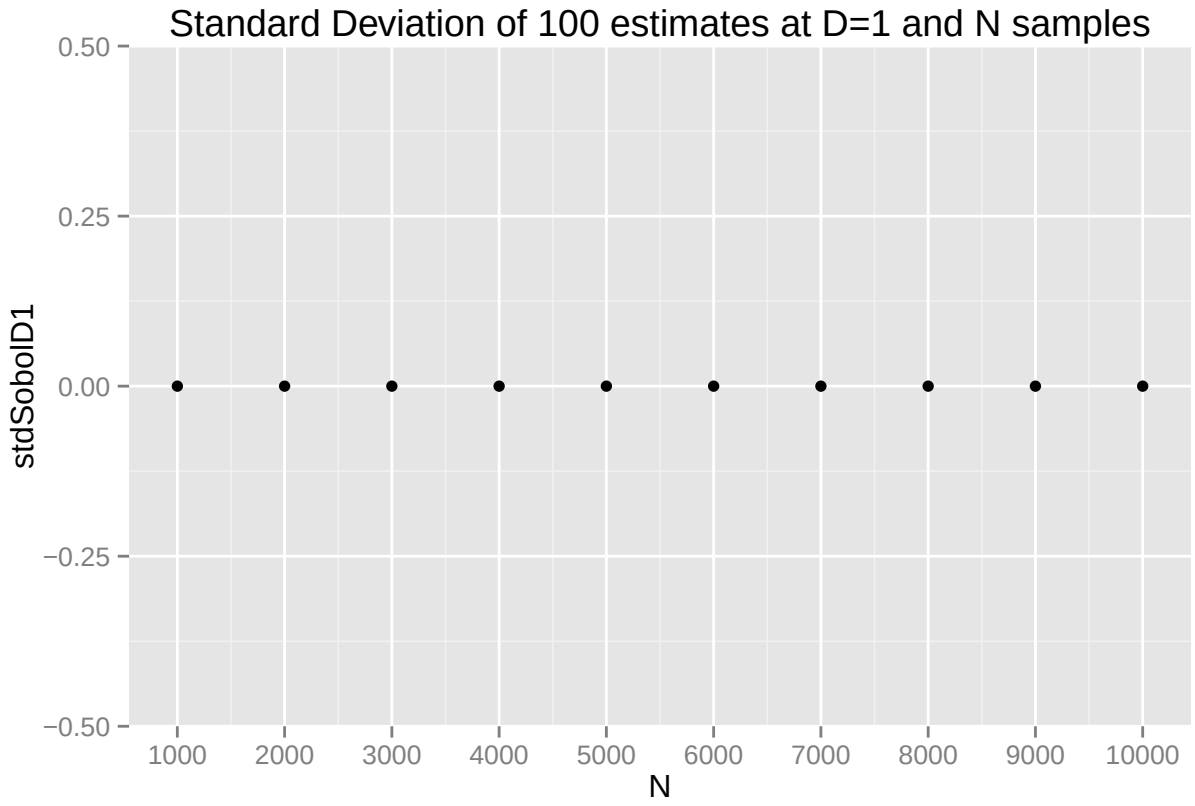
```
EXsobol <- function(D,N){
  c <- numeric(N)
  for (i in 1:N){
    x <- as.vector(sobol(1,2,init=TRUE,scrambling=3,seed=i))
    x <- (x*10)-5 # to scale to (-5,5), multiply by ten and subtract 5
    c[i] <- exp(-0.5*sum(t(x)*x))/((2*pi)^(D/2))
  }
  mean(c)
}

sobolD1 <- matrix(data=NA, nrow=100, ncol=length(N))
colnames(sobolD1) <- N
rownames(sobolD1) <- 1:100
for(i in 1:length(N)){
  for (j in 1:100){
    sobolD1[j,i] <- EXsobol(1,N[i]) # D = 1
  }
}
meanSobolD1 <- apply(sobolD1,2,mean)
stdSobolD1 <- apply(sobolD1,2,sd)
cvSobolD1 <- meanSobolD1/stdSobolD1
sobolD1 <- t(data.frame("mean"=meanSobolD1,"sd"=stdSobolD1,"cv"=cvSobolD1))
summary <- rbind(summary,sobolD1[,1])
summary <- rbind(summary,sobolD1[,4])
kable(sobolD1)
```

	1000	2000	3000	4000	5000	6000	7000	8000	9000	
mean	0.026025	0.0272877	0.0262468	0.0257008	0.0252638	0.0248756	0.0249779	0.0250358	0.0250993	0.02
sd	0.000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.00
cv	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	

```
sobolD1 <- as.data.frame(t(sobolD1))
ggplot(sobolD1) + geom_point(aes(x=N,y=meanSobolD1,label="mean")) +
  geom_hline(yintercept=0.1,color="red") +
  geom_text(aes(2000,0.1,label = "E[c(x)] = 0.1", vjust = -1)) +
  scale_x_continuous(breaks=N) +
  ggtitle("Mean of 100 estimates at D=1 and N samples")
```




```
ggplot(sobolD1) + geom_point(aes(x=N,y=stdSobolD1,label="standard deviation")) +
  scale_x_continuous(breaks=N) +
  ggtitle("Standard Deviation of 100 estimates at D=1 and N samples")
```

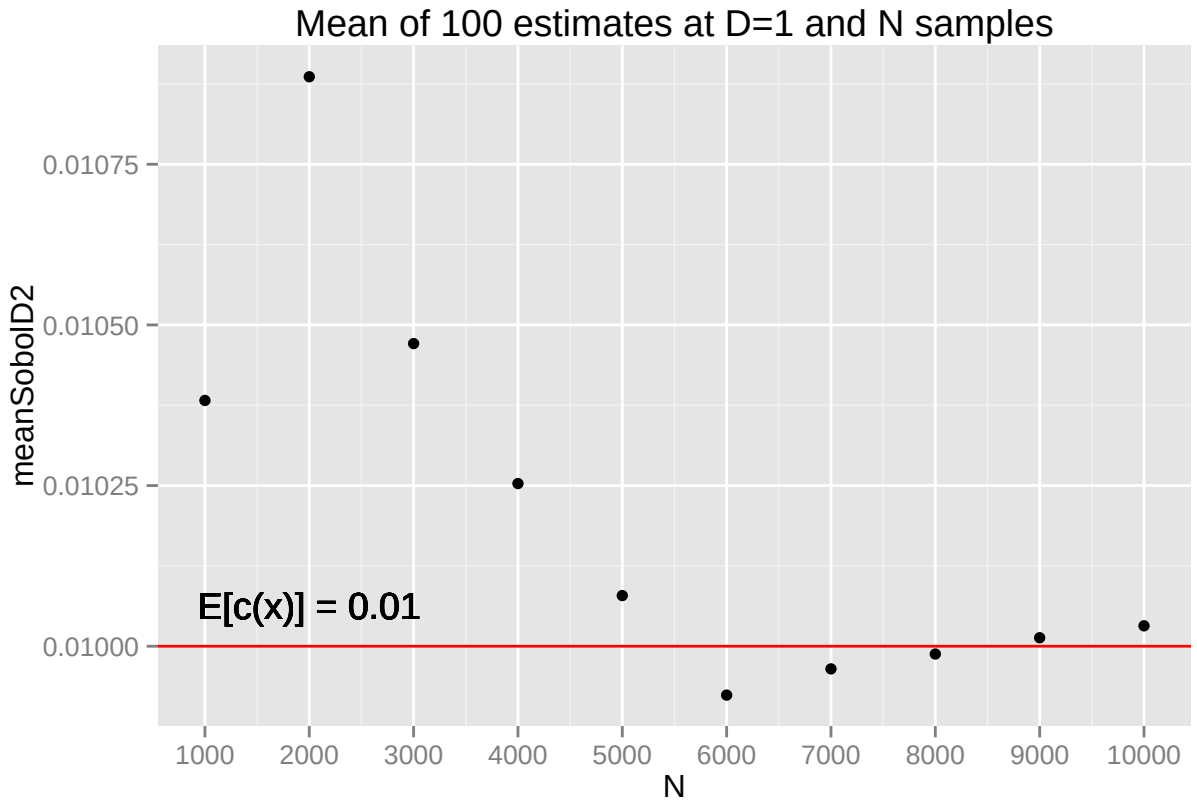



```
sobolD2 <- matrix(data=NA, nrow=100, ncol=length(N))
colnames(sobolD2) <- N
rownames(sobolD2) <- 1:100
for(i in 1:length(N)){
  for (j in 1:100){
    sobolD2[j,i] <- EXsobol(2,N[i]) # D = 2
  }
}
meanSobolD2 <- apply(sobolD2,2,mean)
stdSobolD2 <- apply(sobolD2,2,sd)
cvSobolD2 <- meanSobolD2/stdSobolD2
sobolD2 <- t(data.frame("mean"=meanSobolD2,"sd"=stdSobolD2,"cv"=cvSobolD2))
summary <- rbind(summary,sobolD2[,1])
summary <- rbind(summary,sobolD2[,4])
kable(sobolD2)
```

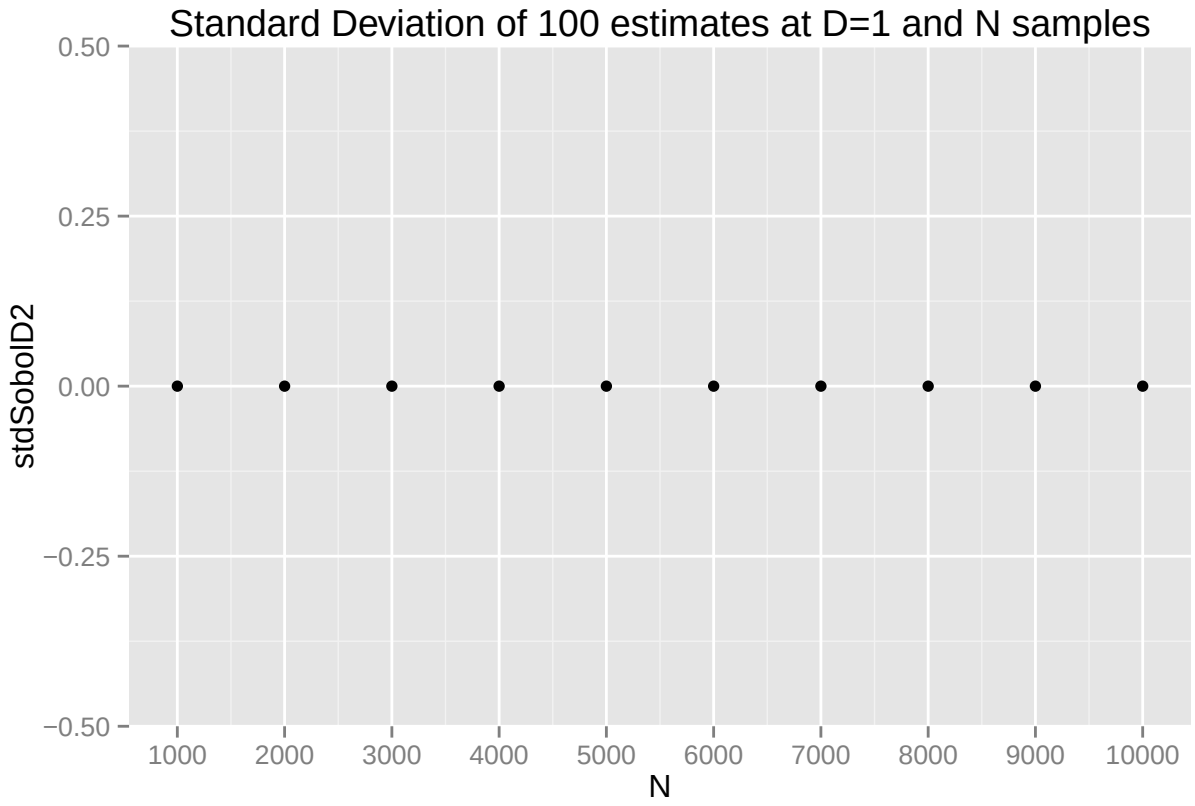
	1000	2000	3000	4000	5000	6000	7000	8000	9000	
mean	0.0103825	0.0108862	0.010471	0.0102531	0.0100788	0.0099239	0.0099648	0.0099878	0.0100132	0.010
sd	0.0000000	0.0000000	0.000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.000
cv	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	

```
sobolD2 <- as.data.frame(t(sobolD2))
ggplot(sobolD2) + geom_point(aes(x=N,y=meanSobolD2,label="mean")) +
  geom_hline(yintercept=0.01,color="red") +
  geom_text(aes(2000,0.01,label = "E[c(x)] = 0.01", vjust = -1)) +
  scale_x_continuous(breaks=N) +
```

```
  ggtitle("Mean of 100 estimates at D=1 and N samples")
```




```
ggplot(sobolD2) + geom_point(aes(x=N,y=stdSobolD2,label="standard deviation")) +
  scale_x_continuous(breaks=N) +
  ggtitle("Standard Deviation of 100 estimates at D=1 and N samples")
```



For D = 1,2 we see zero variance in the estimates at each value of N, but the mean is not close to the analytical value for either dimension.

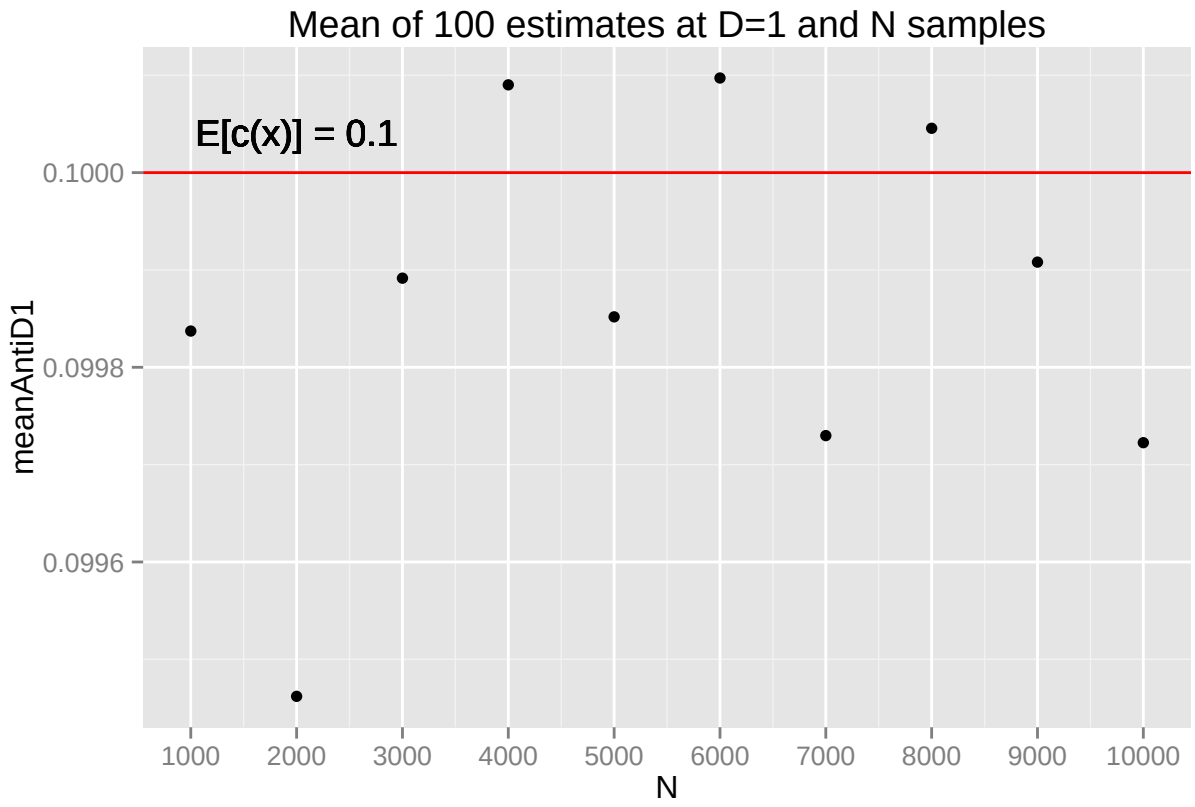
c) Antithetic Variates

```
EXanti <- function(D,N){
  cx1 <- numeric(N/2)
  cx2 <- numeric(N/2)
  for (i in 1:(N/2)){
    x1 <- runif(D,0,1)
    x2 <- 1 - x1
    x1 <- (x1*10)-5
    x2 <- (x2*10)-5
    cx1[i] <- exp(-0.5*sum(t(x1)*x1))/((2*pi)^(D/2))
    cx2[i] <- exp(-0.5*sum(t(x2)*x2))/((2*pi)^(D/2))
    }
  cx <- (mean(cx1)+mean(cx2))/2
  cx
}

antiD1 <- matrix(data=NA, nrow=100, ncol=length(N))
colnames(antiD1) <- N
rownames(antiD1) <- 1:100
for(i in 1:length(N)){
  for (j in 1:100){
    antiD1[j,i] <- EXanti(1,N[i]) # D = 1
  }
```

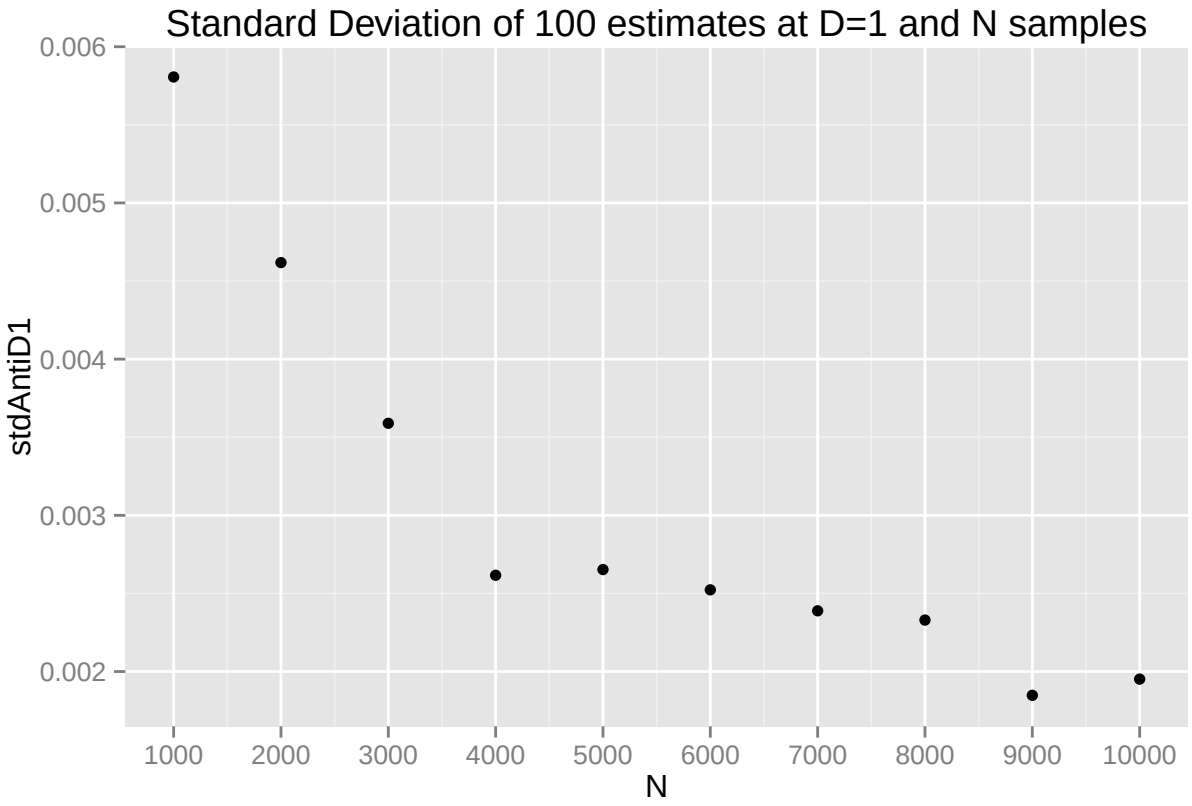
```
}
meanAntiD1 <- apply(antiD1,2,mean)
stdAntiD1 <- apply(antiD1,2,sd)
cvAntiD1 <- meanAntiD1/stdAntiD1
antiD1 <- t(data.frame("mean"=meanAntiD1,"sd"=stdAntiD1,"cv"=cvAntiD1))
summary <- rbind(summary,antiD1[,1])
summary <- rbind(summary,antiD1[,4])
kable(antiD1)
```

	1000	2000	3000	4000	5000	6000	7000	8000	
mean	0.0998372	0.0994620	0.0998916	0.1000902	0.0998518	0.1000972	0.0997298	0.1000455	0.099
sd	0.0058063	0.0046181	0.0035894	0.0026163	0.0026530	0.0025231	0.0023888	0.0023295	0.001
cv	17.1946690	21.5375233	27.8298870	38.2569938	37.6373549	39.6729943	41.7489704	42.9468339	54.071

```
antiD1 <- as.data.frame(t(antiD1))
ggplot(antiD1) + geom_point(aes(x=N,y=meanAntiD1,label="mean")) +
  geom_hline(yintercept=0.1,color="red") +
  geom_text(aes(2000,0.1,label = "E[c(x)] = 0.1", vjust = -1)) +
  scale_x_continuous(breaks=N) +
  ggtitle("Mean of 100 estimates at D=1 and N samples")
```




```
ggplot(antiD1) + geom_point(aes(x=N,y=stdAntiD1,label="standard deviation")) +
  scale_x_continuous(breaks=N) +
  ggtitle("Standard Deviation of 100 estimates at D=1 and N samples")
```

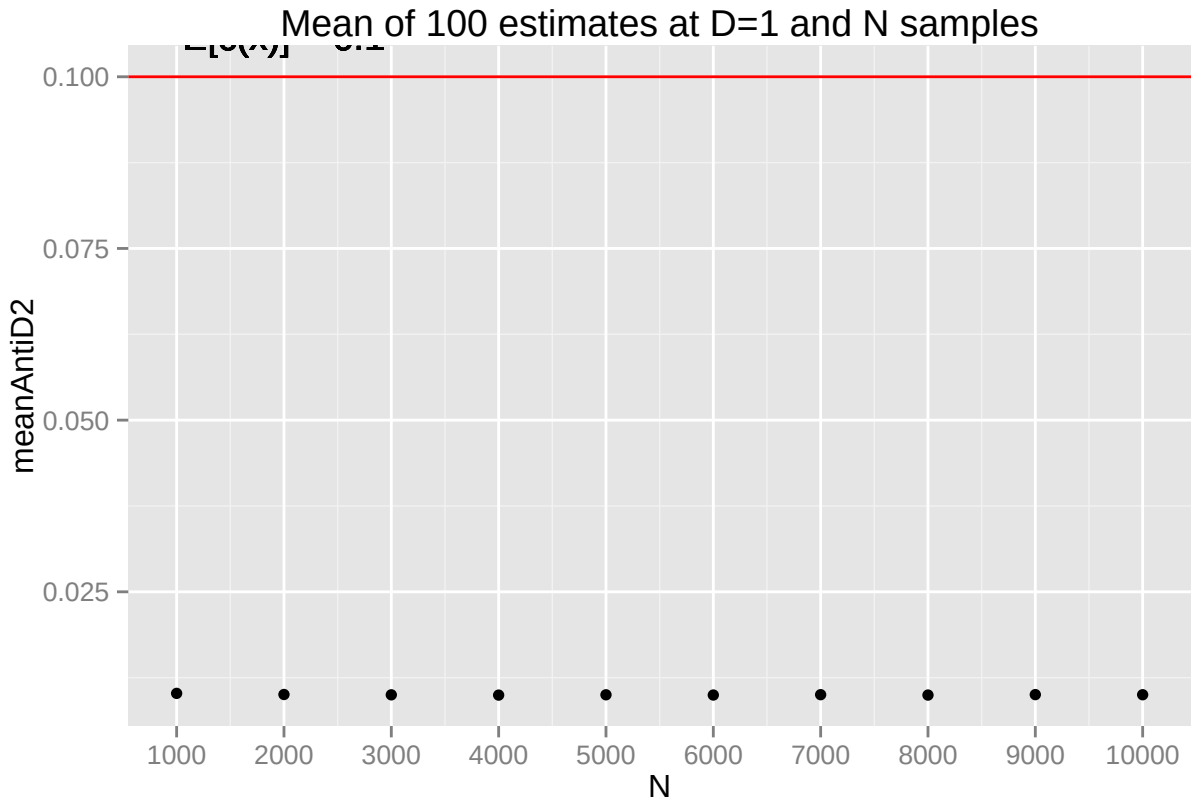



```
antiD2 <- matrix(data=NA, nrow=100, ncol=length(N))
colnames(antiD2) <- N
rownames(antiD2) <- 1:100
for(i in 1:length(N)){
  for (j in 1:100){
    antiD2[j,i] <- EXanti(2,N[i]) # D = 2
  }
}
meanAntiD2 <- apply(antiD2,2,mean)
stdAntiD2 <- apply(antiD2,2,sd)
cvAntiD2 <- meanAntiD2/stdAntiD2
antiD2 <- t(data.frame("mean"=meanAntiD2,"sd"=stdAntiD2,"cv"=cvAntiD2))
summary <- rbind(summary,antiD2[,1])
summary <- rbind(summary,antiD2[,4])
kable(antiD2)
```

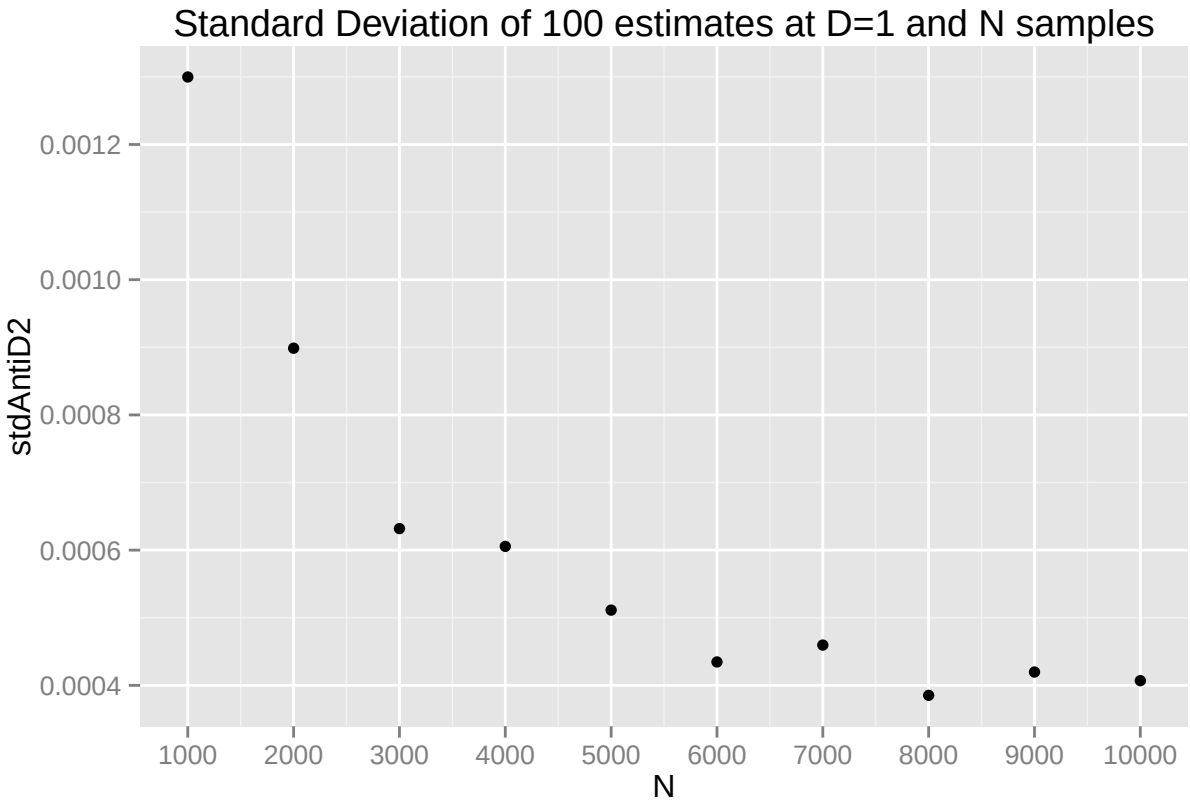
	1000	2000	3000	4000	5000	6000	7000	8000	9
mean	0.0102126	0.0100497	0.0100004	0.0099603	0.0100059	0.0099710	0.0100234	0.0099656	0.0100
sd	0.0012998	0.0008987	0.0006319	0.0006055	0.0005113	0.0004346	0.0004596	0.0003852	0.0004
cv	7.8568842	11.1827600	15.8265605	16.4490646	19.5704132	22.9416262	21.8095605	25.8689656	23.9025

```
antiD2 <- as.data.frame(t(antiD2))
ggplot(antiD2) + geom_point(aes(x=N,y=meanAntiD2,label="mean")) +
  geom_hline(yintercept=0.1,color="red") +
  geom_text(aes(2000,0.1,label = "E[c(x)] = 0.1", vjust = -1)) +
  scale_x_continuous(breaks=N) +
```

```
  ggtitle("Mean of 100 estimates at D=1 and N samples")
```




```
ggplot(antiD2) + geom_point(aes(x=N,y=stdAntiD2,label="standard deviation")) +
  scale_x_continuous(breaks=N) +
  ggtitle("Standard Deviation of 100 estimates at D=1 and N samples")
```



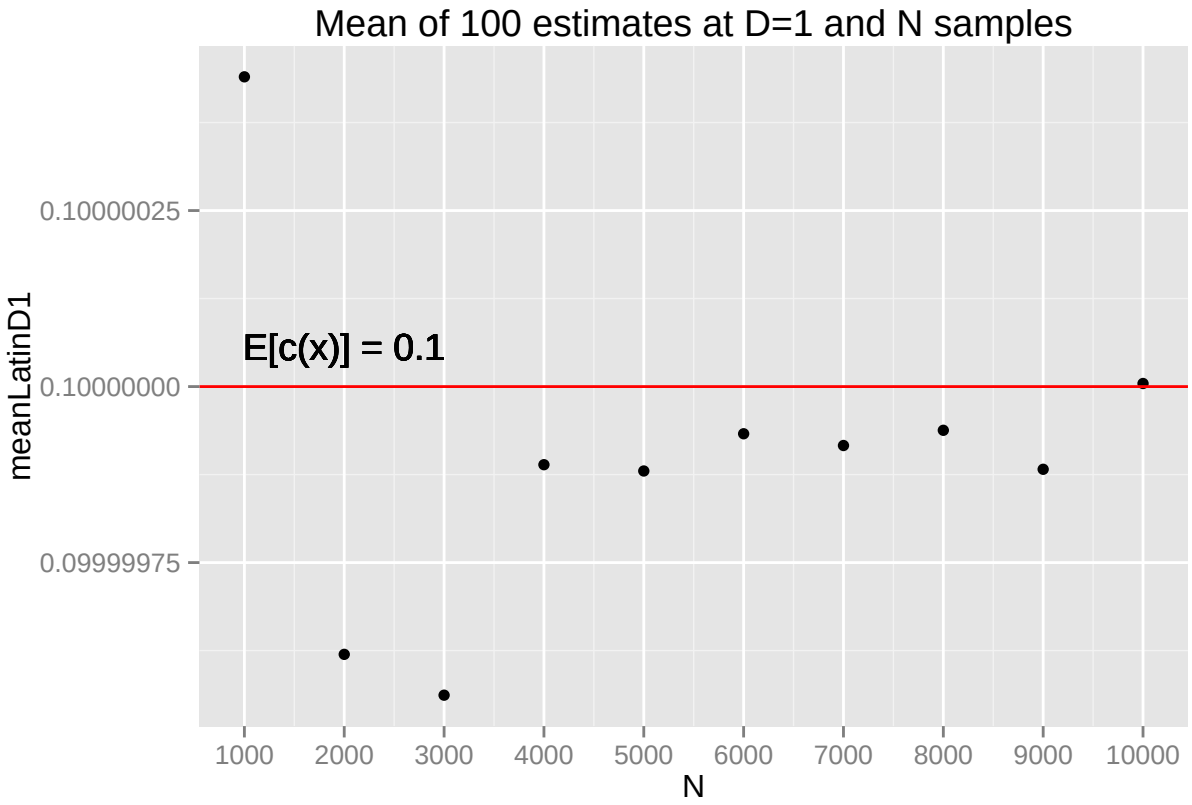
d) Latin Hypercube Sampling

```
library(lhs)
EXlatin <- function(D,N){
  c <- numeric(N)
  x <- randomLHS(N,D)
  x <- (x*10)-5
  if (D==1){for (i in 1:N){c[i] <- exp(-0.5*sum(t(x[i])*x[i]))/((2*pi)^(D/2))}}
  else if (D > 1){for (i in 1:N){c[i] <- exp(-0.5*sum(t(x[i,])*x[i,]))/((2*pi)^(D/2))}}
  mean(c)
}

latinD1 <- matrix(data=NA, nrow=100, ncol=length(N))
colnames(latinD1) <- N
rownames(latinD1) <- 1:100
for(i in 1:length(N)){
  for (j in 1:100){
    latinD1[j,i] <- EXlatin(1,N[i]) # D = 1
  }
}
meanLatinD1 <- apply(latinD1,2,mean)
stdLatinD1 <- apply(latinD1,2,sd)
cvLatinD1 <- meanLatinD1/stdLatinD1
latinD1 <- t(data.frame("mean"=meanLatinD1,"sd"=stdLatinD1,"cv"=cvLatinD1))
summary <- rbind(summary,latinD1[,1])
summary <- rbind(summary,latinD1[,4])
```

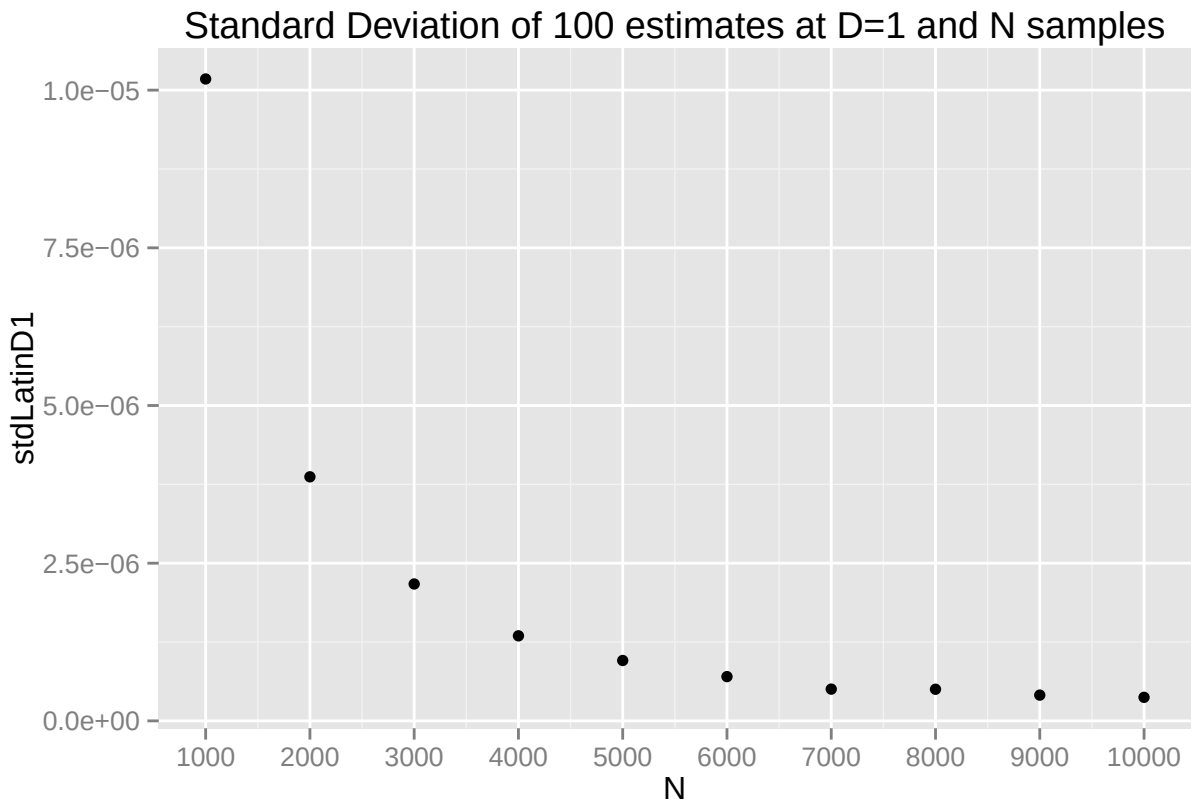
```
kable(latinD1)
```

	1000	2000	3000	4000	5000	6000	7000	
mean	0.1000004	9.99996e-02	9.999960e-02	9.999990e-02	9.999990e-02	9.999990e-02	9.999990e-02	9
sd	0.0000102	3.90000e-06	2.200000e-06	1.300000e-06	1.000000e-06	7.000000e-07	5.000000e-07	5
cv	9826.0362845	2.58437e+04	4.607653e+04	7.417423e+04	1.044703e+05	1.425642e+05	1.986313e+05	1.9

```
latinD1 <- as.data.frame(t(latinD1))
ggplot(latinD1) + geom_point(aes(x=N,y=meanLatinD1,label="mean")) +
  geom_hline(yintercept=0.1,color="red") +
  geom_text(aes(2000,0.1,label = "E[c(x)] = 0.1", vjust = -1)) +
  scale_x_continuous(breaks=N) +
  ggtitle("Mean of 100 estimates at D=1 and N samples")
```




```
ggplot(latinD1) + geom_point(aes(x=N,y=stdLatinD1,label="standard deviation")) +
  scale_x_continuous(breaks=N) +
  ggtitle("Standard Deviation of 100 estimates at D=1 and N samples")
```

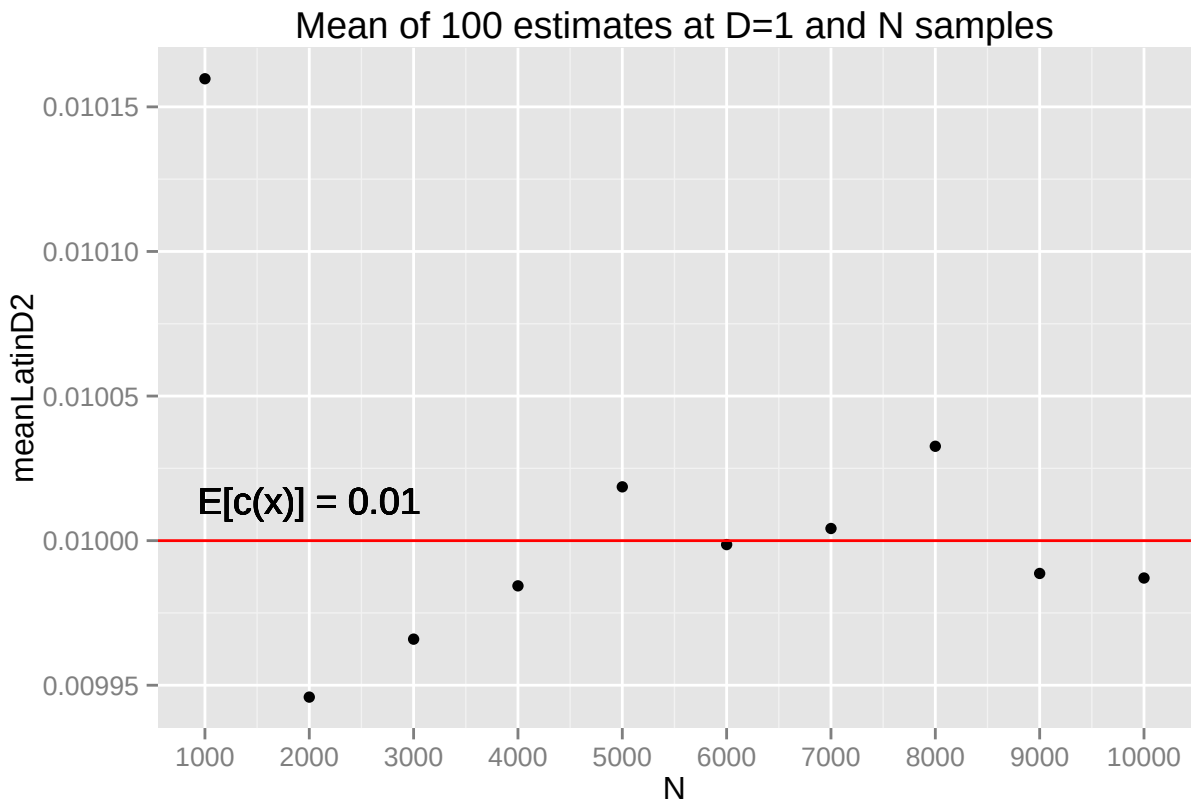



```
latinD2 <- matrix(data=NA, nrow=100, ncol=length(N))
colnames(latinD2) <- N
rownames(latinD2) <- 1:100
for(i in 1:length(N)){
  for (j in 1:100){
    latinD2[j,i] <- EXlatin(2,N[i]) # D = 2
  }
}
meanLatinD2 <- apply(latinD2,2,mean)
stdLatinD2 <- apply(latinD2,2,sd)
cvLatinD2 <- meanLatinD2/stdLatinD2
latinD2 <- t(data.frame("mean"=meanLatinD2,"sd"=stdLatinD2,"cv"=cvLatinD2))
summary <- rbind(summary,latinD2[,1])
summary <- rbind(summary,latinD2[,4])
kable(latinD2)
```

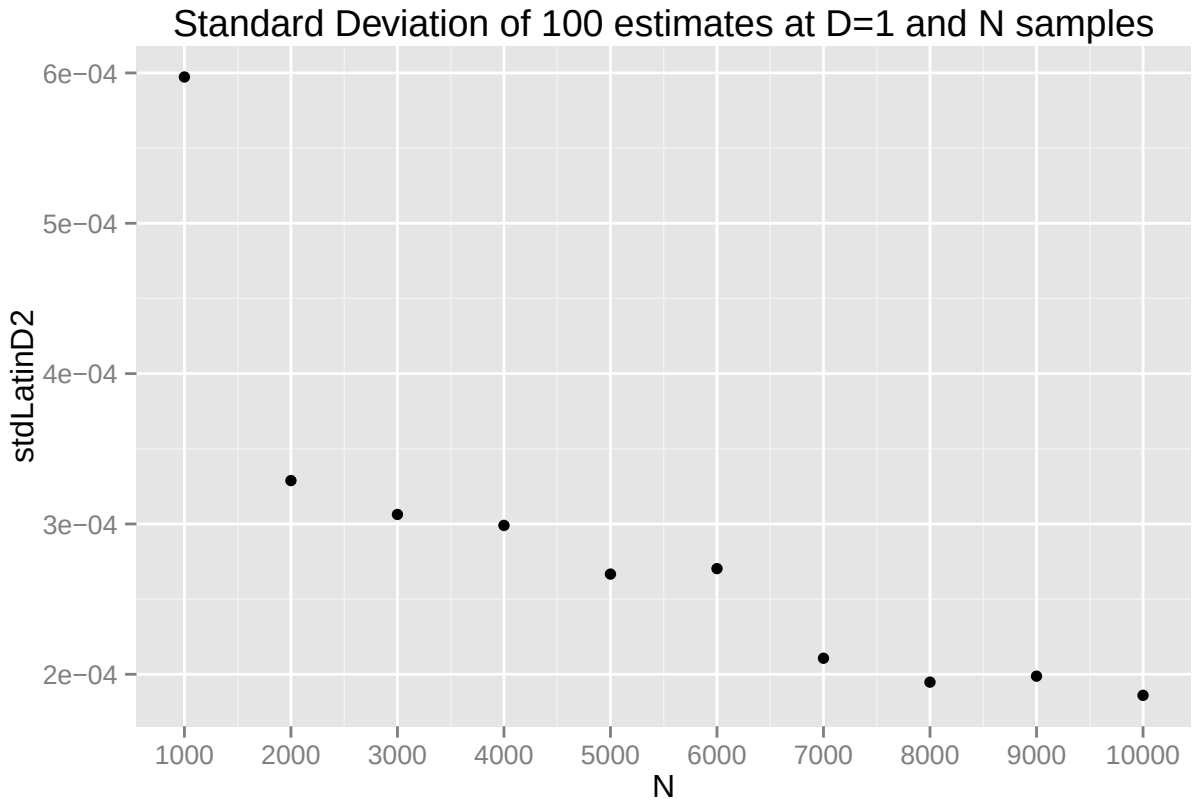
	1000	2000	3000	4000	5000	6000	7000	8000	
mean	0.0101597	0.0099459	0.0099659	0.0099844	0.0100186	0.0099986	0.0100042	0.0100326	0.009
sd	0.0005973	0.0003288	0.0003063	0.0002990	0.0002667	0.0002703	0.0002107	0.0001948	0.000
cv	17.0091271	30.2451928	32.5337029	33.3882427	37.5716774	36.9910921	47.4846584	51.5070211	50.253

```
latinD2 <- as.data.frame(t(latinD2))
ggplot(latinD2) + geom_point(aes(x=N,y=meanLatinD2,label="mean")) +
  geom_hline(yintercept=0.01,color="red") +
  geom_text(aes(2000,0.01,label = "E[c(x)] = 0.01", vjust = -1)) +
  scale_x_continuous(breaks=N) +
```

```
  ggtitle("Mean of 100 estimates at D=1 and N samples")
```

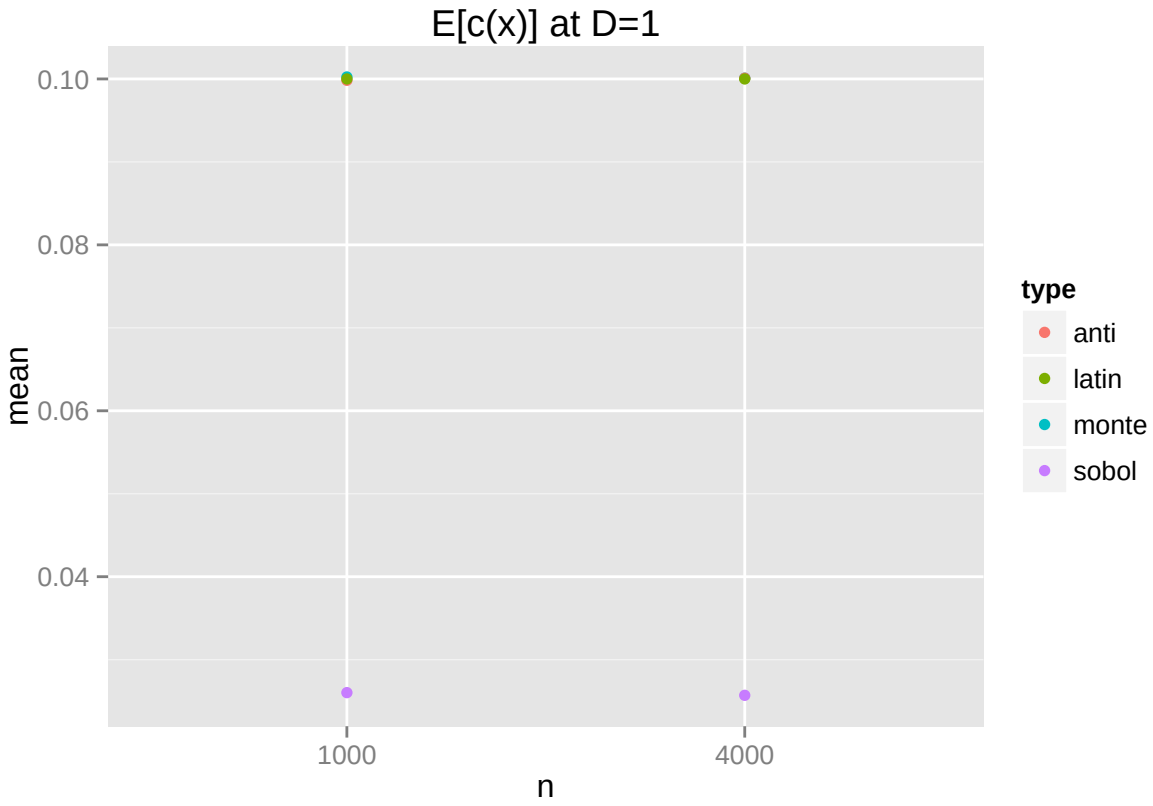



```
ggplot(latinD2) + geom_point(aes(x=N,y=stdLatinD2,label="standard deviation")) +
  scale_x_continuous(breaks=N) +
  ggtitle("Standard Deviation of 100 estimates at D=1 and N samples")
```

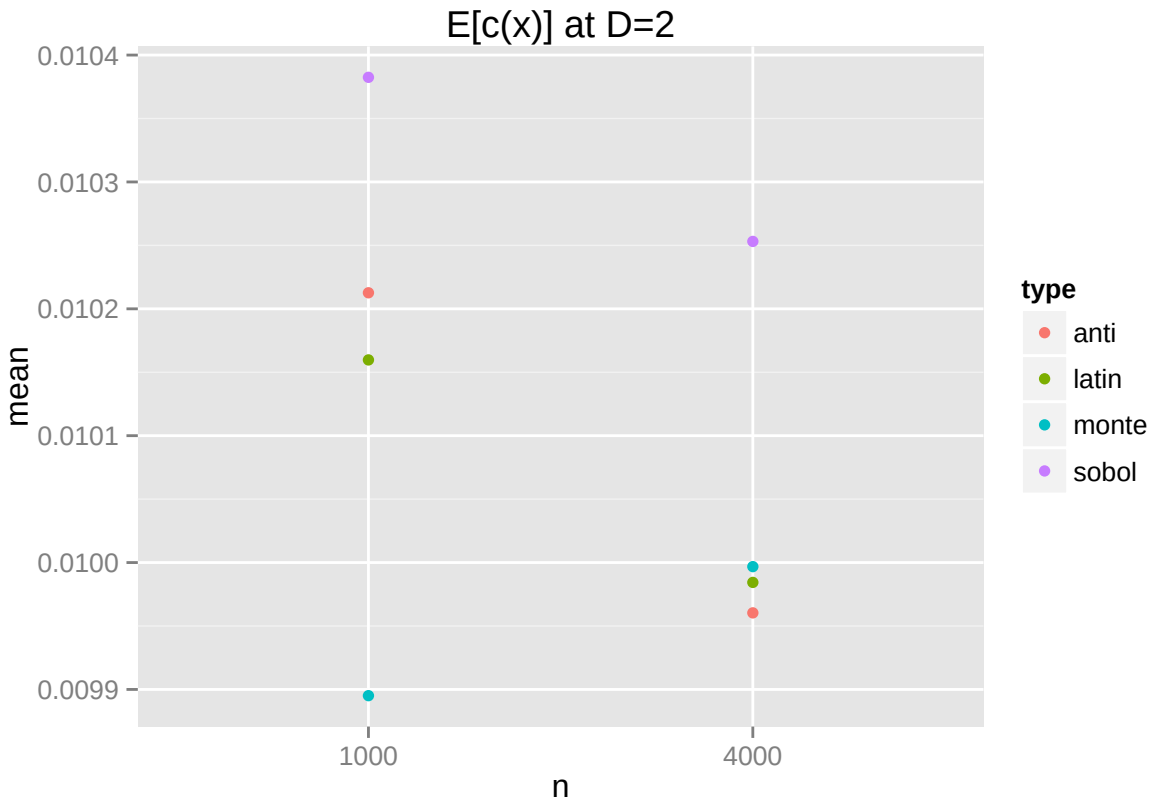


f) Summary

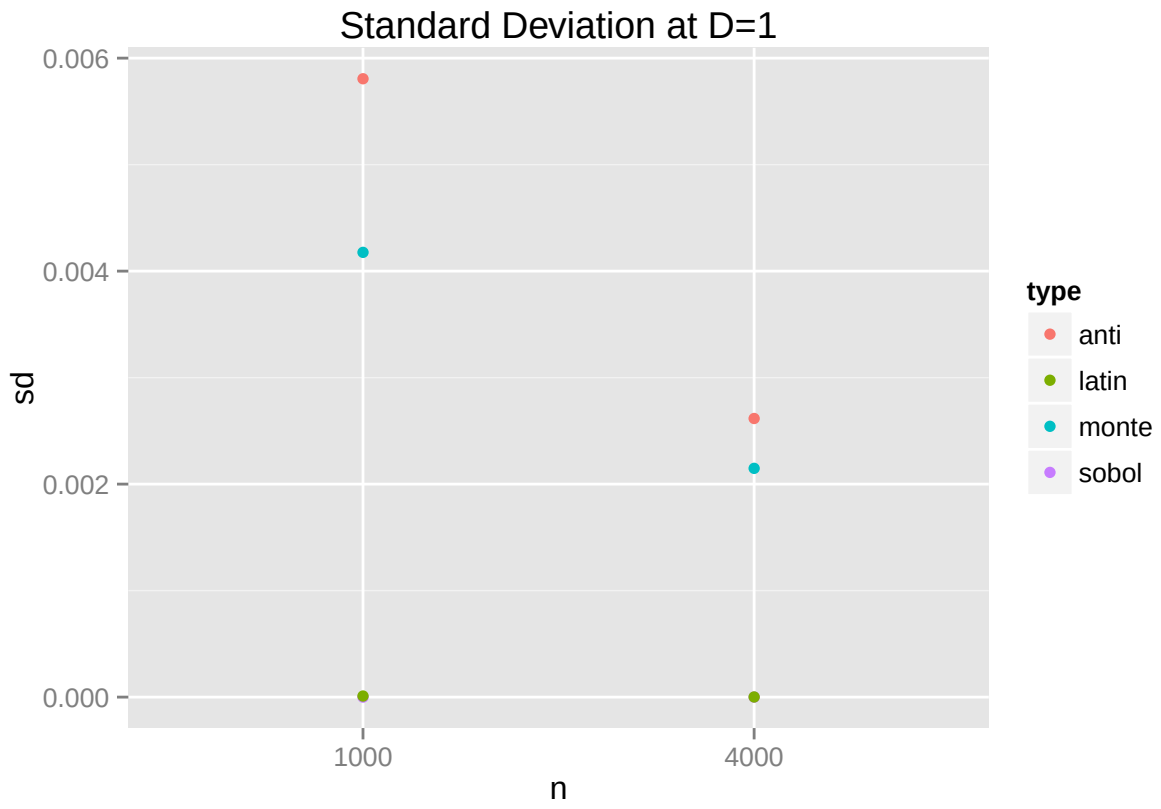
```
colnames(summary) <- c("mean","sd","cv")
summary["n"] <- rep(c("1000","4000"),8)
summary["type"] <- c(rep(c("monte"),4),rep(c("sobol"),4),rep(c("anti"),4),rep(c("latin"),4))
summary["d"] <- rep(c(1,1,2,2),4)
D1frame <- subset(summary,d==1)
D2frame <- subset(summary,d==2)
ggplot(D1frame,aes(x=n,y = mean,color=type)) + geom_point() + ggtitle("E[c(x)] at D=1")
```



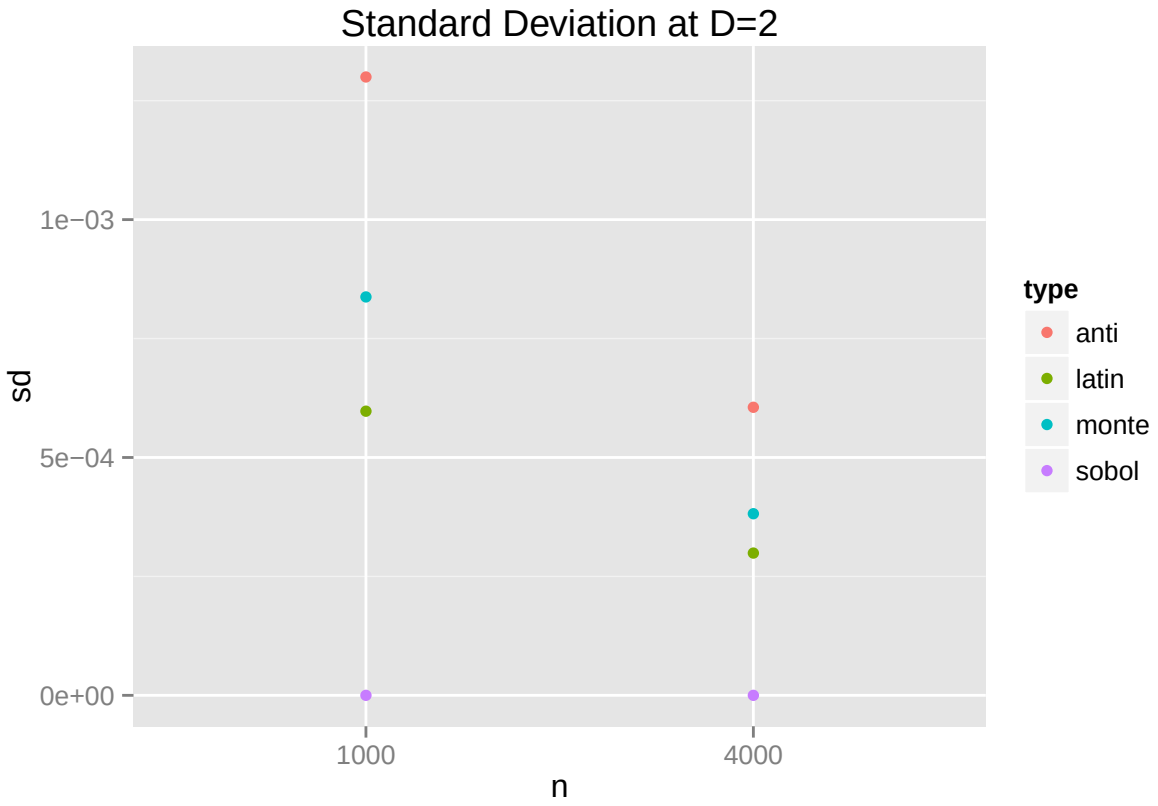

```
ggplot(D2frame,aes(x=n,y = mean,color=type)) + geom_point() + ggtitle("E[c(x)] at D=2")
```



```
ggplot(D1frame,aes(x=n,y = sd,color=type)) + geom_point() + ggtitle("Standard Deviation at D=1")
```




```
ggplot(D2frame,aes(x=n,y = sd,color=type)) + geom_point() + ggtitle("Standard Deviation at D=2")
```



It appears that all of the methods, save Sobol, closely approximate the analytical value of $E[c(x)] = 0.1$ at $D = 1$. At the same time, the antithetic approach and Monte Carlo approach had the highest variation, but did decrease significantly from $n = 1000$ to $n = 4000$. At $D = 2$, the Monte Carlo method was closesly to the analytical value of $E[c(x)] = 0.01$, although it demonstrated higher variation that the Latin Hypercube and Sobol methods. The variance reduction methods seem strong for all of the methods, although Sobol seems to demonstrate the least variation. All of the methods show rapidly decreasing variation as the value of N increases.

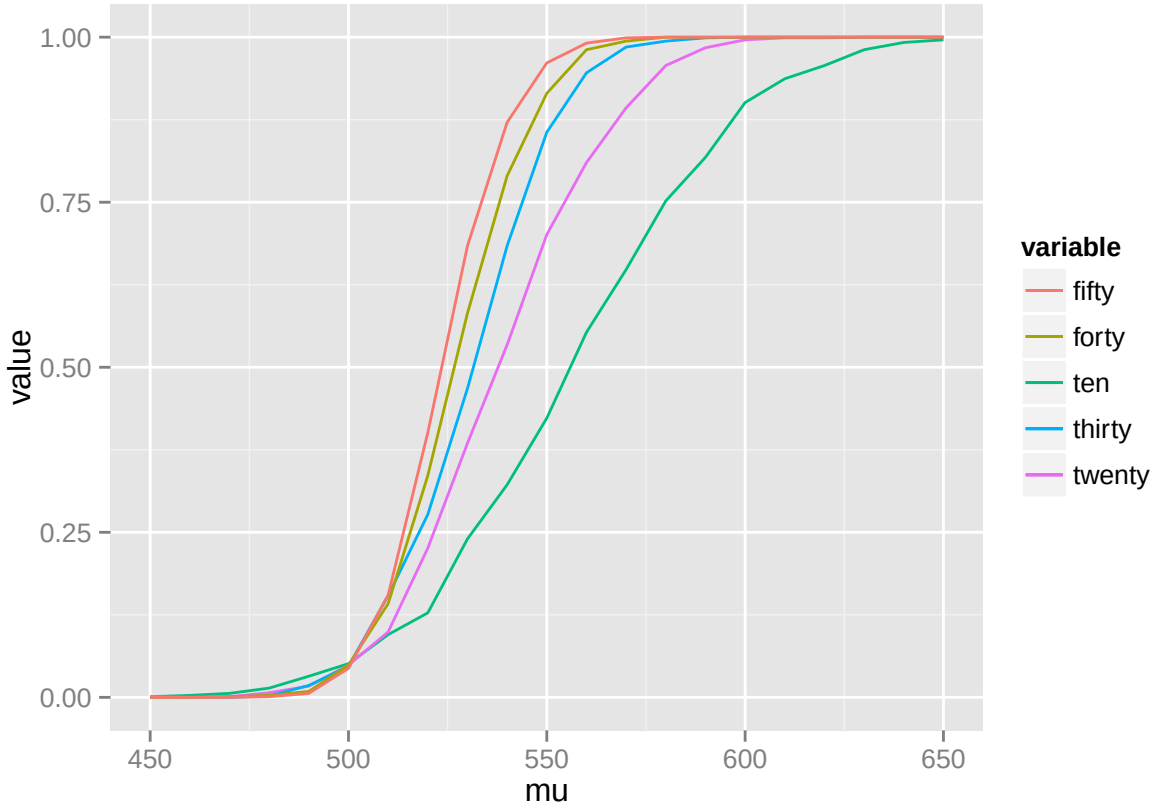
Exercises from *Statistical Computing with R*

6.3

Plot the power curves for the t-test in Example 6.9 for sample sizes 10, 20, 30, 40, and 50, but omit the standard error bars. Plot the curves on the same graph, each in a different color or different line type, and include a legend. Comment on the relation between power and sample size.

```
powercurve <- function(n){
m <- 1000
mu0 <- 500
sigma <- 100
mu <- c(seq(450, 650, 10)) #alternatives
M <- length(mu)
power <- numeric(M)
  for (i in 1:M) {
    mu1 <- mu[i]
    pvalues <- replicate(m, expr = {
  #simulate under alternative mu1
      x <- rnorm(n, mean = mu1, sd = sigma)
```

```
      ttest <- t.test(x,
      alternative = "greater", mu = mu0)
      ttest$p.value } )
  power[i] <- mean(pvalues <= .05)
  }
 power
}
mu <- c(seq(450, 650, 10))
powerCurves <- data.frame(mu,"ten"=powercurve(10),"twenty"=powercurve(20),"thirty"=powercurve(30),"fort
ggplot(powerCurves,aes(x=mu, y = value, color = variable)) +
  geom_line(aes(x=mu,y=ten,colour="ten")) +
  geom_line(aes(x=mu,y=twenty,colour="twenty")) +
  geom_line(aes(x=mu,y=thirty,colour="thirty")) +
  geom_line(aes(x=mu,y=forty,colour="forty")) +
  geom_line(aes(x=mu,y=fifty,colour="fifty"))
```



The power curves are all similarly shaped - for each of the sample size the power of the curve, or the rejecting probability of rejecting H_O given that the true value of the parameter is Θ, is quite low fpr $mu < 500$. When $mu > 500$, the power increases more rapidly, the larger the sample size becomes. If we prefer to reduce Type II error, then we would want to pick the curve with the highest power, which is based on a sample size of 50.

6.4

Suppose that $X_1, ..., X_n$ are a random sample from a lognormal distribution with unknown parameters. Construct a 95% confidence interval for the parameter μ. Use a Monte Carlo method to obtain an empirical estimate of the confidence level.

7.1

Compute a jackknife estimate of the bias and the standard error of the correlation statistic in Example 7.2.

Jackknife estimate of bias:

```
library(bootstrap)
n <- nrow(law)
lsat <- law$LSAT
gpa <- law$GPA
theta.hat <- mean(lsat) / mean(gpa)
print (theta.hat)
```

```
## [1] 1.939681
```

```
theta.jack <- numeric(n)
for (i in 1:n)
theta.jack[i] <- mean(lsat[-i]) / mean(gpa[-i])
bias <- (n - 1) * (mean(theta.jack) - theta.hat)
print(bias) #jackknife estimate of bias
```

```
## [1] 0.0002506089
```

Jackkknife estimate of standard error:

```
se <- sqrt((n-1) *
mean((theta.jack - mean(theta.jack))^2))
print(se)
```

```
## [1] 0.02525517
```

7.4

Refer to the air-conditioning data set aircondit provided in the boot package. The 12 observations are the times in hours between failures of airconditioning equipment [63, Example 1.1]:

$$3, 5, 7, 18, 43, 85, 91, 98, 100, 130, 230, 487$$

Assume that the times between failures follow an exponential model $Exp(\lambda)$. Obtain the MLE of the hazard rate λ and use bootstrap to estimate the bias and standard error of the estimate.

To get the MLE of the failure rate in the exponential distribution, we can divide the number of failures by the total elapsed time.

$$\lambda = \frac{r}{\sum_{i=1}^{r} t_i + (n-r)T}$$

Our MLE for λ is:

```
library(boot)
length(aircondit$hours)/sum(aircondit$hours)
```

```
## [1] 0.00925212
```

Our bootstraph estimate for the standard error of the estimate is:

```
B <- 200
n <- 12
R <- numeric(B)
for (b in 1:B) {
i <- sample(1:n, size = n, replace = TRUE)
hours <- aircondit$hours[i]
R[b] <- length(hours)/sum(hours)
}
print(se.R <- sd(R))
```

```
## [1] 0.005446482
```

Our bootstraph estimate for the bias of the estimate is:

```
theta.hat <- 0.00925212
B <- 2000
n <- 12
theta.b <- numeric(B)
for (b in 1:B) {
i <- sample(1:n, size = n, replace = TRUE)
hours <- aircondit$hours[i]
theta.b[b] <- length(hours)/sum(hours)
}
bias <- mean(theta.b - theta.hat)
bias
```

```
## [1] 0.001353937
```