Stat 413 Assignment 2

Milagros N. Cortez

2023-03-13

1. Volume of a d-ball

In d-dimensional space, let $\mathcal{R}_d = [-1, 1]^n$ be a hypercube and $\mathcal{B}_d = \{x = (x_1, \ldots, x_n) : x_1^2 + \ldots + x_n^2 \leq 1\}$ be a d-dimensional ball. Use random sampling from $\mathcal{R}_d$ to estimate the d-volume of the ball. Consider sampling after re-scaling the Beta distribution to be on $[-1, 1]$

Before continuing on the problem, we first note that the $d-$volume of a ball of radius $r = 1$ is:

$$d - volume(V_S) = \frac{\pi^{d/2} r^d}{\Gamma(d/2+1)} = \frac{\pi^{d/2}}{\Gamma(d/2+1)}$$

We also create a function to calculate the true d-volume of a ball of radius r:

```
# Function to calculate true volume of the d-ball with an r radius
true_dar = function(d, r){
  vol = (pi^(d/2)*(r^d))/gamma((d/2)+1)
  return(vol)
}
```

At $d = 4$, the 4-volume of a ball with radius $r = 1$ is:

$$4 - volume(V_S) = \frac{\pi^{4/2}}{\Gamma(4/2+1)} = \frac{\pi^2}{\Gamma(3)} = \frac{\pi^2}{2}$$

Using the function above we find this value to be approximately:

```
d4_vol = true_dar(4, 1)
d4_vol
```

```
## [1] 4.934802
```

At $d = 100$, the 100-volume of a ball with radius $r = 1$ is: $100 - volume(V_S) = \frac{\pi^{100/2}}{\Gamma(100/2+1)} = \frac{\pi^{50}}{\Gamma(51)} = \frac{\pi^{50}}{50!}$

Using the function above we find this value to be approximately:

```
d100_vol = true_dar(100, 1)
d100_vol
```

```
## [1] 2.368202e-40
```

Given that $\mathcal{R}_d = [-1, 1]^n$, we have for $d = 1$ the interval $[-1, 1]$ which is of length 2. Moreover, the volume of a hypercube would be:

$$V_C = 2^d$$

Which is an exponential function. At $d = 4$ we get a volume of $V_C = 2^4 = 16$, and at $d = 100$ we get a volume of $V_C = 2^{100}$ which we can also see computed in the table below:

```
hypercvol = function(d){2^d}
inf = matrix(c(hypercvol(4), hypercvol(100)), ncol =1, byrow = TRUE)
colnames(inf) = c("Volume of Hypercube")
rownames(inf) = c("d=4", "d=100")
info = as.table(inf)
knitr::kable(head(inf), "simple")
```

	Volume of Hypercube
d=4	1.600000e+01
d=100	1.267651e+30

From the table we note that as d increases, the volume of the hypercube (exponential function) increases rapidly while the d-volume of the ball decreases.

Monte Carlo Volume Estimation

For the following cases we proceed to use Monte Carlo to estimate the volume of the d-ball which can be done by sampling uniform and non-uniform densities.

When sampling a with a uniform density, as in the first case of this problem, we generate d-dimensional vectors uniformly within a hypercube $\mathcal{R}_d$ by filling entries in X with iid Uniform[-1, 1] random variabes. Sampling in this way, we estimate the mean and variance of the volume of the d-ball in the following way:

mean: $\hat{\mu} = \frac{2^d}{n} \sum_{i=1}^{n} \phi(x_i)$

variance: $\hat{\sigma}^2 = \frac{2^{2d}\hat{\mu}(1-\hat{\mu})}{n-1}$

Where $\phi(x_i) = 1$ if $x \in \mathcal{B}_d$, and 0 otherwise.

Sampling with a non-uniform density on the hypercube $\mathcal{R}_d$ on the other hand, we apply importance sampling which consists of having a random variable X with a density function $f(x)$, such that $f(x) > 0$ on $\{x : g(x) > 0\}$ where $g(x)$ is the average value of a function over an interval (a, b). The estimate of $E[g(X)/f(X)]$ by Monte Carlo integration therefore consists on computing the average:

$\frac{1}{n} \sum_{i=1}^{n} \frac{g(X_i)}{f(X_i)}$

where the random variables $X_1, \ldots, X_n$ are generated from the distribution of $f(x)$ which is the importance function. We also note that the variance of $g(X)/f(X)$ is $Var(g(X)/f(X))/n$ which is expected to be small if $g(X)/f(X)$ is nearly constant which indicates $f(.)$ is approximately close to $g(x)$. With this we also note that $f(.)$ should be easy to simulte to obtain a result, and when sampling iid $x_1, \ldots, x_n \sim f(x)$. The condition $f(x) > 0$ is set in place so that that $f(x)$ is bounded away from zero and we don't divide by zero when calculating the volume.

Thus, from the above we obtain the volume and variance of the volume of the d-ball in the following way:

$Vol(\mathcal{B}_d) \approx \frac{2^d}{n} \sum_{i=1}^{n} \frac{g(x_i)}{f(x_i)}$

$Var(\mathcal{B}_d) \approx \frac{2^{2d}}{n} Var(\frac{g(x_i)}{f(x_i)})$

1. Sampling each $x_i \overset{\text{iid}}{\sim} Uniform[-1,1]$

We first set the boolean function for the ball with $r = 1$, a function for uniform sampling with Monte Carlo, and the function for 95% error bounds:

```r
# Circle function
phi.circle = function(x, r=1){
  return(sum(x^2)<= r^2)
}

# Monte Carlo Uniform Sampling
VMCU = function(num = 500, d, phi){
  mcGrid = matrix(runif(n = num*d, min = -1, max = 1), nrow= num, ncol=d)
  in_region = apply(mcGrid, 1, phi)
  muh = (sum(in_region)/num)
  volume = 2^(d)*muh # Volume estimation
  varh = (muh*(1-muh))/(num-1)
  avar = 2^(2*d)*varh # variance estimation
  return(c(volume, avar))
}

# 95% error bounds
# Use of Z distribution as sample size n is large
vol_CI = function(muh, varh, alpha){
  norm = c(muh-qnorm(1-alpha/2)*sqrt(varh),  muh+qnorm(1-alpha/2)*sqrt(varh))
  return(rbind(norm))
}
```

Trying this for a small d, we try to estimate the volume of a ball $d = 4$ with a sample size $n = 100000$ and we get:

```r
n = 100000
calc1 = VMCU(num = n, 4, phi = phi.circle)
inf = matrix(c(calc1[1], d4_vol ,calc1[2], vol_CI(calc1[1], calc1[2], 0.05)[1], vol_CI(calc1[1],
calc1[2], 0.05)[2]), ncol =1, byrow = TRUE)
colnames(inf) = c("Value")
rownames(inf) = c("Estimated volume:", "True volume:", "Variance:","Lower bound:", "Upper boun
d:")
info = as.table(inf)
knitr::kable(inf, "simple")
```

	Value
Estimated volume:	4.9104000
True volume:	4.9348022

	Value
Variance:	0.0005445
Lower bound:	4.8646631
Upper bound:	4.9561369

We can see from the output above that the estimated volume is quite close to the true volume of the 4-ball, we also have a small variance. We can also see that the 95% error bounds calculated above captures both the estimated and true volume of the 4-ball.

Trying this for a large d, $d = 100$, we get:

```
n = 100000
calc = VMCU(num = n, 100, phi = phi.circle)
inf = matrix(c(calc[1], d100_vol ,calc[2]), ncol =1, byrow = TRUE)
colnames(inf) = c("Value")
rownames(inf) = c("Estimated Volume:", "True Volume:", "Variance:")
info = as.table(inf)
knitr::kable(head(inf), "simple")
```

	Value
Estimated Volume:	0
True Volume:	0
Variance:	0

We note that when d=100 we get true and estimated values that are approximately close to zero. This is expected since as mentioned above, as d increases the volume of the hypercube increases exponentially while the ball inside decreases.

We try to estimate the volume with a value of d=15 which is just larger than 4 and less than 100, and we can see that for a sample of n=100000:

```
n = 100000
calc2 = VMCU(num = n, 15, phi = phi.circle)
d15_vol = true_dar(15, 1)
inf = matrix(c(calc2[1],  d15_vol, calc2[2], vol_CI(calc2[1], calc2[2], 0.05)[1], vol_CI(calc2
[1], calc2[2], 0.05)[2]), ncol =1, byrow = TRUE)
colnames(inf) = c("Value")
rownames(inf) = c("Estimated Volume:", "True Volume:", "Variance:","Lower bound:", "Upper boun
d:")
info = as.table(inf)
knitr::kable(head(inf), "simple")
```

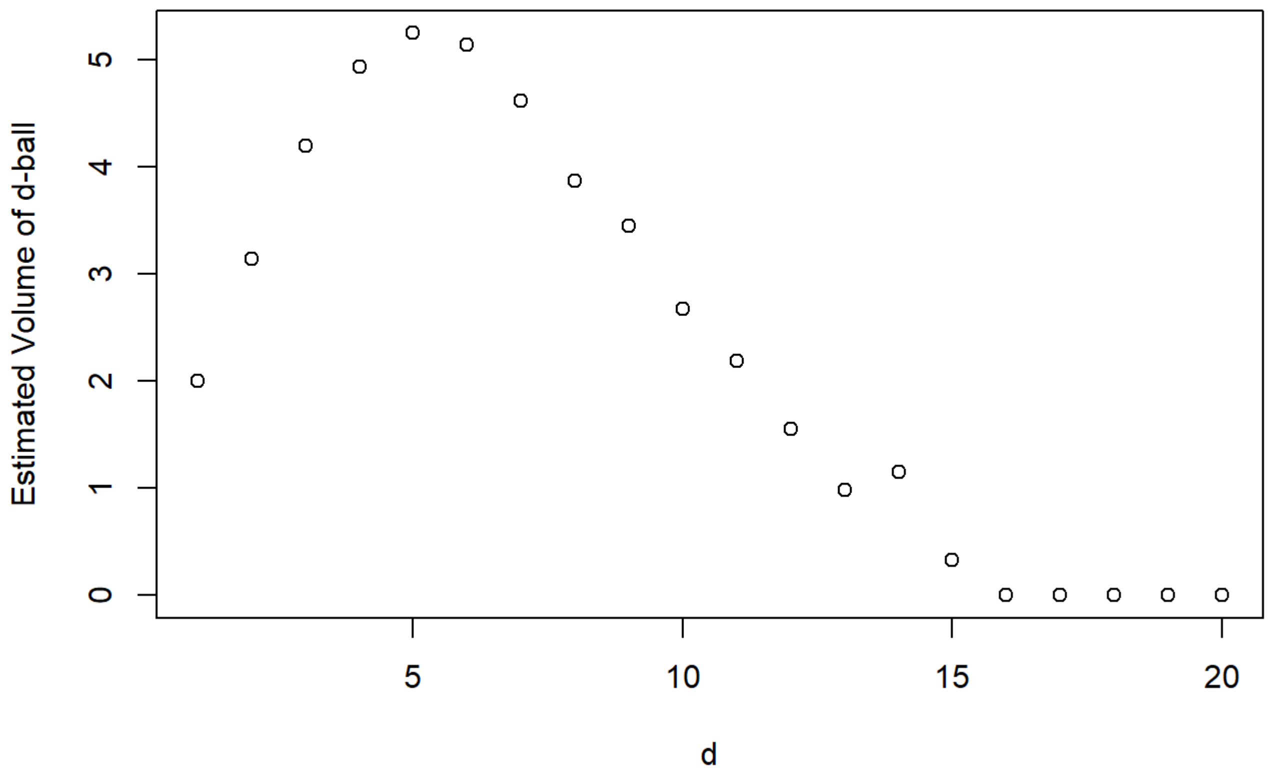
	Value
Estimated Volume:	0.3276800
True Volume:	0.3814433

	Value
Variance:	0.1073742
Lower bound:	-0.3145610
Upper bound:	0.9699210

We can see from the output above that the estimated Volume is also close to the true Volume of the 15-ball. The variance is larger compared to the variance obtained for the 4-ball, which is expected since as d increases, the volume of the hypercube increases, but the volume of the d-ball decreases making it harder to sample and estimate. We can also see that the 95% error bounds captures both the true and estimated volume.

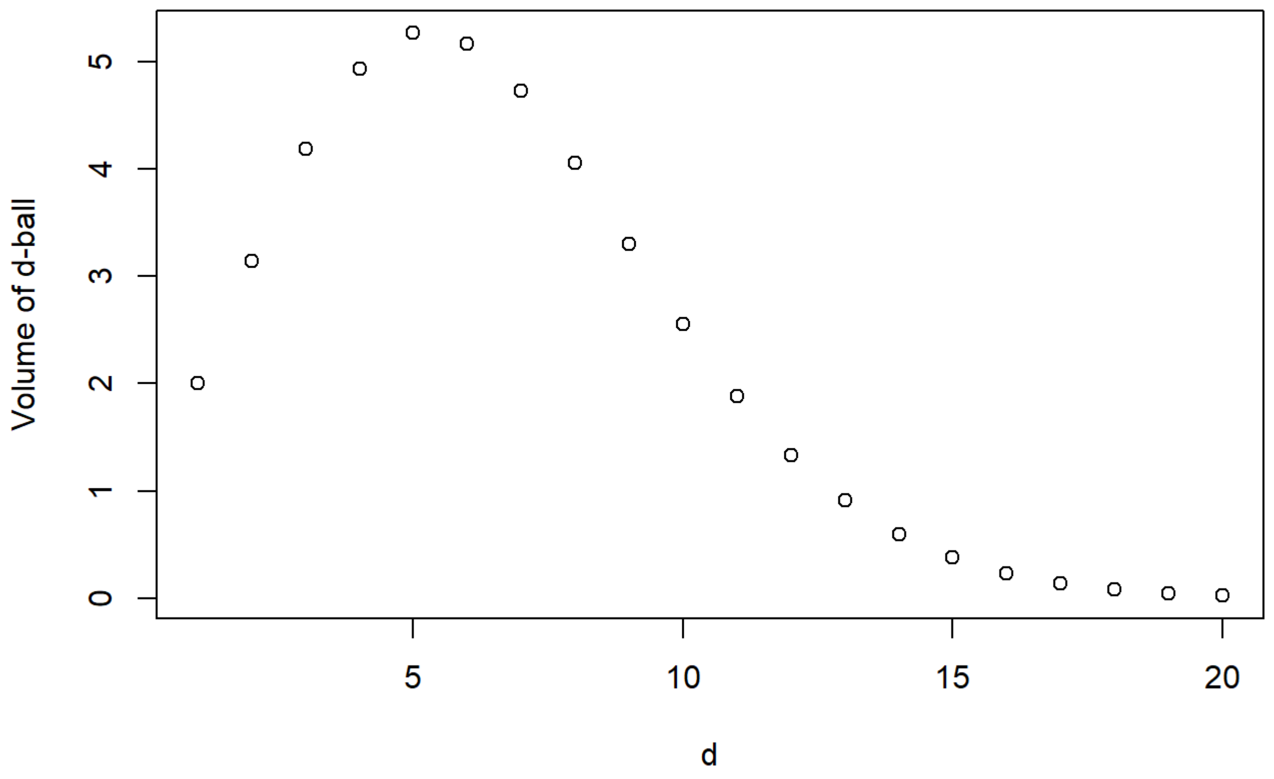
From the outputs above it would be expected that the volume of the d-ball would decrease monotonically as d increases. However, this is not the case. Plotting values of d (from 1 to 20) along with their calculated estimated volume, we get:

```r
x = seq(1,20, 1)
esty = c()
for (i in x){
  est = VMCU(num = 100000, i, phi = phi.circle)[1]
  esty = c(esty, est)
}
plot(x, esty, xlab = "d", ylab = "Estimated Volume of d-ball")
```



Which is close to the plot of values of d (from 1 to 20) along with their true volume:

```
x = seq(1,20, 1)
esty = c()
for (i in x){
  est = true_dar(i, 1)
  esty = c(esty, est)
}
plot(x, esty, xlab = "d", ylab = "Volume of d-ball")
```



We can see from the plots above that the volume of the d-ball steadily increases until $d = 5$ and then decreases to the point that at d=20 and higher d values will have a d-ball volume approximately close to zero.

We proceed to use d=15 for the rest of the problems in question 1.

2. Sampling each $x_i \overset{\text{iid}}{\sim} Beta(1/2, 1/2)$

We first have to re-scale the beta distribution to be on the interval $[-1, 1]$. A beta random variable x defined on the interval $[0, 1]$ can be re-scaled and shifted to obtain a beta random variable on the interval $[a, b] \to [-1, 1]$ by setting $y = x(b - a) + a = x(1 - (-1)) + (-1) = 2x - 1$,

This adjusts the PDF to be:

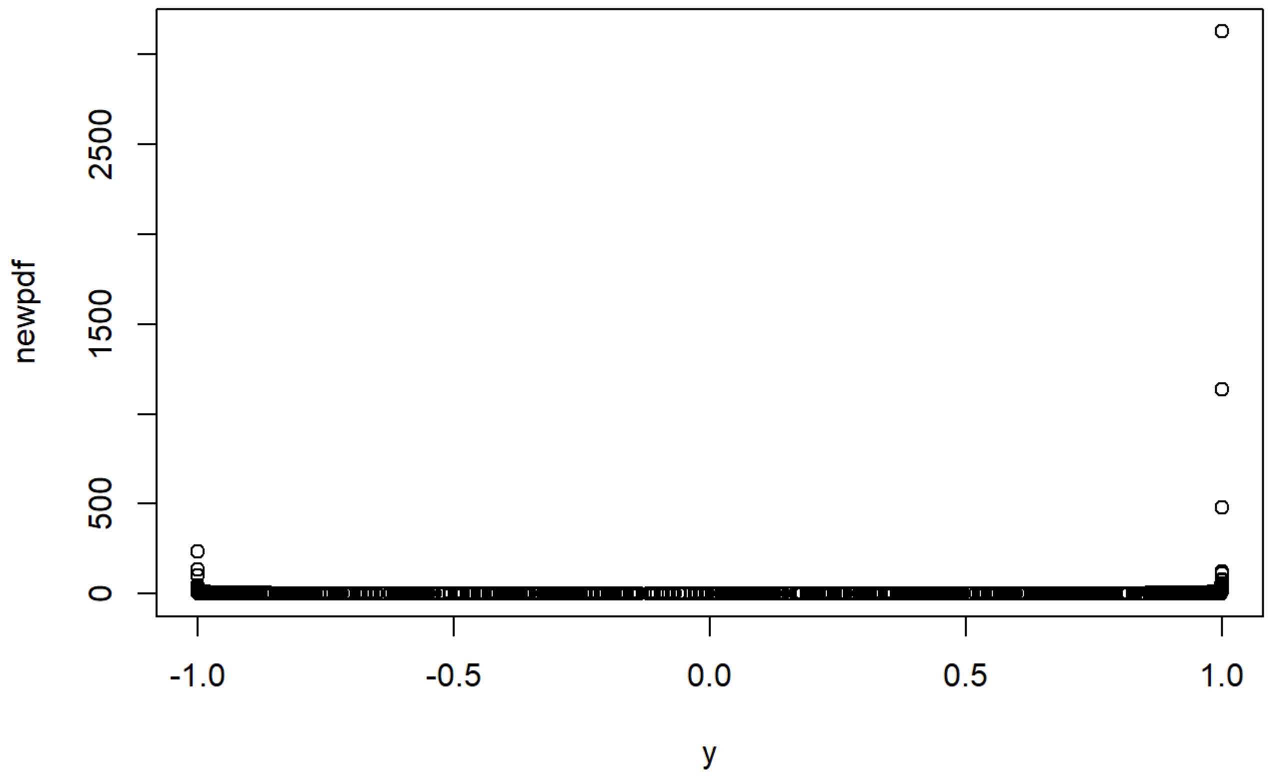
$f(y) = \frac{(y-(a))^{\alpha-1} \times (b-y)^{\beta-1}}{(b-(a))^{\alpha+\beta-2} \times Beta(\alpha,\beta)}$

$$= \frac{(y-(-1))^{\frac{1}{2}-1} \times (1-y)^{\frac{1}{2}-1}}{(1-(-1))^{\frac{1}{2}+\frac{1}{2}-2} \times Beta(\frac{1}{2}, \frac{1}{2})}$$

$$= \frac{(y+1)^{-\frac{1}{2}} \times (1-y)^{-\frac{1}{2}}}{2^{-1} \times Beta(\frac{1}{2}, \frac{1}{2})}$$

We can see the shifted distribution in the plot below:

```
y = 2*rbeta(1000, 0.5, 0.5)-1
newpdf = ((y+1)^(-0.5)*(1-y)^(-0.5))/2^(-1)*rbeta(1000, 0.5, 0.5)
plot(y, newpdf)
```



Given that the distribution is not uniform, we proceed to work with the case of Monte Carlo estimation sampling with non-uniform distributions. We apply the shifted beta distribution $y = 2x - 1$ with $x \sim Beta(1/2, 1/2)$:

```r
# Monte Carlo sampling with transformed Beta function
VMCB = function(n, d, nu, phi){
  mcGrid = matrix(2*rbeta(n*d, nu, nu)-1, nrow= n, ncol=d)
  in_region = apply(mcGrid, 1, phi)

  weight1 = dbeta((mcGrid+1)/2, shape1 =nu, shape2 = nu) #set the weights
  weightn = apply(weight1, 1, prod)
  # calculating the volume of the ball
  muh = 2^(d)*(mean(in_region/abs(weightn)))
  # calculating the variance of the ball
  varh = 2^(2*d)*var(in_region/abs(weightn))/n
  return(c(muh, varh))
}
```

Estimating the volume of a $d = 4$ ball, we get:

```r
n = 10000
nucalc1 = VMCB(n, 4, 0.5,phi = phi.circle)
inf = matrix(c(nucalc1[1], d4_vol ,nucalc1[2], vol_CI(nucalc1[1], nucalc1[2], 0.05)[1], vol_CI(n
ucalc1[1], nucalc1[2], 0.05)[2]), ncol =1, byrow = TRUE)
colnames(inf) = c("Value")
rownames(inf) = c("Estimated Volume:", "True Volume:", "Variance:","Lower bound:", "Upper boun
d:")
info = as.table(inf)
knitr::kable(head(inf), "simple")
```

	Value
Estimated Volume:	4.6849942
True Volume:	4.9348022
Variance:	0.0276091
Lower bound:	4.3593267
Upper bound:	5.0106617

We can see that the estimated volume is close to the true volume, and both the estimated value and true value are within the 95% error bounds. We can also see that the calculated variance is significantly larger than the calculated variance found through uniform sampling.

Trying this for a large d, $d = 15$, we get:

```
n = 100000
nucalc2 = VMCB(n, 15, 0.5, phi = phi.circle)
inf = matrix(c(nucalc2[1], d15_vol ,nucalc2[2], vol_CI(nucalc2[1], nucalc2[2], 0.05)[1], vol_CI
(nucalc2[1], nucalc2[2], 0.05)[2]), ncol =1, byrow = TRUE)
colnames(inf) = c("Value")
rownames(inf) = c("Estimated Volume:", "True Volume:", "Variance:","Lower bound:", "Upper boun
d:")
info = as.table(inf)
knitr::kable(head(inf), "simple")
```

	Value
Estimated Volume:	0.0000000
True Volume:	0.3814433
Variance:	0.0000000
Lower bound:	0.0000000
Upper bound:	0.0000000

We can see from the output above that when sampling from $Beta(1/2, 1/2)$ the estimated volume and variance of the $d = 15$ ball is approximately 0. This is possibly due to the points in the $Beta(1/2, 1/2)$ distribution being at the edges of the interval, compared to the 15-ball which lies at the center of the hyper cube $\mathcal{R}_{15}$. In addition almost all of the volume in a hyper cube is away from the center, and mostly in the corners. This makes it harder to sample and estimate the volume of the $d = 15$ ball.

3. Sampling each $x_i \overset{\text{iid}}{\sim} Beta(2, 2)$

We first have to re-scale the beta distribution to be on the interval $[-1, 1]$. A beta random variable x defined on the interval $[0, 1]$ can be re-scaled and shifted to obtain a beta random variable on the interval $[a, b] \to [-1, 1]$ by setting $y = x(b - a) + a = x(1 - (-1)) + (-1) = 2x - 1$.

This adjusts the PDF to be:

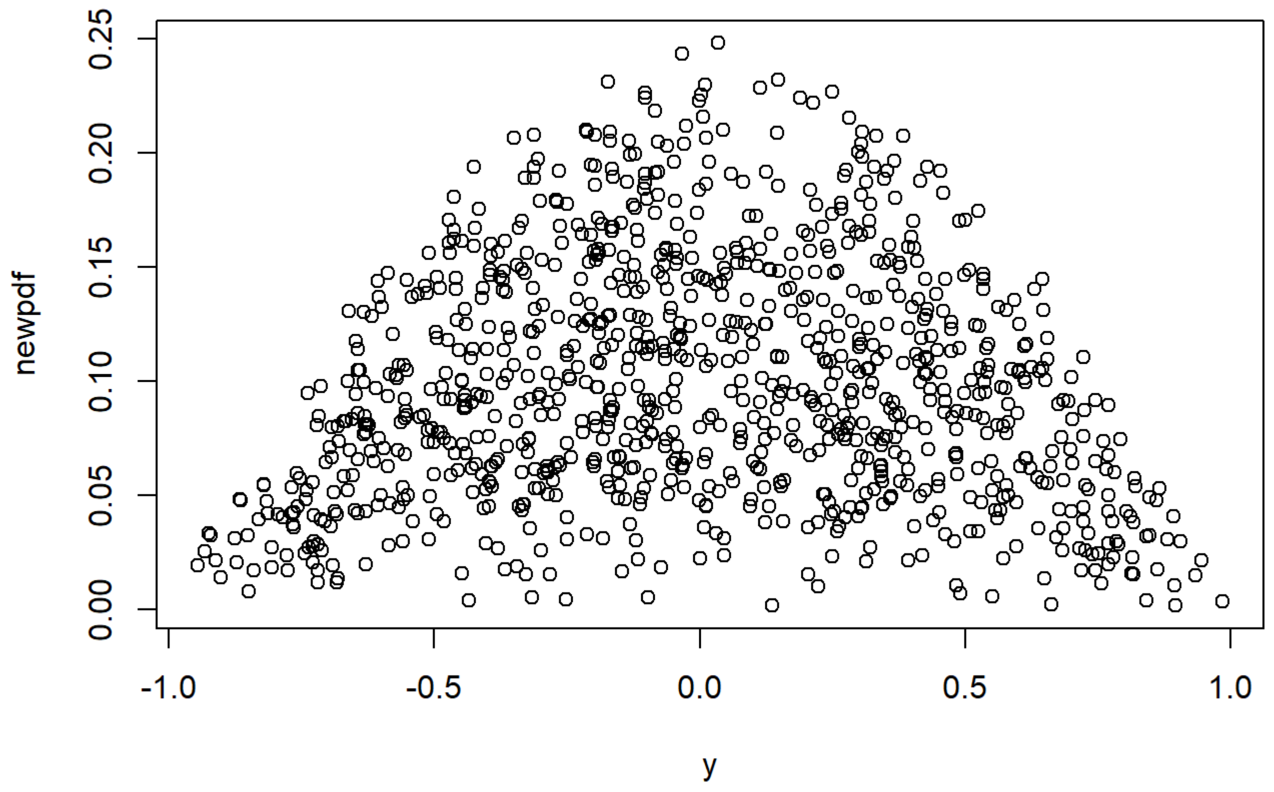
$$f(y) = \frac{(y-(a))^{\alpha-1} \times (b-y)^{\beta-1}}{(b-(a))^{\alpha+\beta-2} \times Beta(\alpha,\beta)}$$

$$= \frac{(y-(-1))^{2-1} \times (1-y)^{2-1}}{(1-(-1))^{2+2-2} \times Beta(2,2)}$$

$$= \frac{(y+1) \times (1-y)}{2^2 \times Beta(2,2)}$$

We can see the shifted distribution in the plot below:

```
y = 2*rbeta(1000, 2, 2)-1
newpdf = ((y+1)^(1)*(1-y)^(1))/2^(2)*rbeta(1000, 2, 2)
plot(y, newpdf)
```



Estimating the volume of a $d = 4$ ball, we get:

```
n = 10000
nucalc3 = VMCB(n, 4, 2,phi = phi.circle)
inf = matrix(c(nucalc3[1], d4_vol ,nucalc3[2], vol_CI(nucalc3[1], nucalc3[2], 0.05)[1], vol_CI(n
ucalc3[1], nucalc3[2], 0.05)[2]), ncol =1, byrow = TRUE)
colnames(inf) = c("Value")
rownames(inf) = c("Estimated Volume:", "True Volume:", "Variance:","Lower bound:", "Upper boun
d:")
info = as.table(inf)
knitr::kable(head(inf), "simple")
```

	Value
Estimated Volume:	4.9034801
True Volume:	4.9348022
Variance:	0.0016974
Lower bound:	4.8227304
Upper bound:	4.9842299

From the output above we can see that the estimated volume is approximately close to the true volume and both lie within the 95% error bounds. We can also see that the variance is slightly larger than the variance calculated through uniform sampling.

Trying this for a large d, $d = 15$, we get:

```
n = 10000
nucalc4 = VMCB(n, 15, 2 ,phi = phi.circle)
inf = matrix(c(nucalc4[1], d15_vol ,nucalc4[2], vol_CI(nucalc4[1], nucalc4[2], 0.05)[1], vol_CI
(nucalc4[1], nucalc4[2], 0.05)[2]), ncol =1, byrow = TRUE)
colnames(inf) = c("Value")
rownames(inf) = c("Estimated Volume:", "True Volume:", "Variance:", "Lower bound:", "Upper boun
d:")
info = as.table(inf)
knitr::kable(head(inf), "simple")
```

	Value
Estimated Volume:	0.5441365
True Volume:	0.3814433
Variance:	0.0110020
Lower bound:	0.3385548
Upper bound:	0.7497183

From the output above we can see that the estimated volume is close to the true volume and both values lie within the 95% error bounds. We can also see that the variance is significantly smaller than the variance calculated through uniform sampling.

Conclusion

Overall, we can see that estimation differs between large and small dimensions. When estimating for a small dimension such as d=4, we get an estimate value that is approximately close to the true value when sampling from the three distributions above. The variance however changes depending on the sampling distribution used. When sampling from a $uniform[-1, 1]$ distribution, we obtained a smaller variance, followed by the variance found when sampling from a $Beta(2, 2)$ distribution, with the highest variance being the one found when sampling with a $Beta(1/2, 1/2)$. All 95% error bounds captured the true and expected volumes.

When estimating for a larger dimension, we first found that at d=100 the estimated volume is approximately 0 which is expected as the true value is also approximately zero, and that generally as d increases the volume of the d-ball decreases after d=5, while the volume of the hyper-cube increases exponentially as d increases. At d=15 we observed that when sampling from a uniform distribution we got an expected volume approximately close to the true volume. However, when sampling from this distribution a couple times we also noted that the estimated volume varied from the true volume. When sampling from $Beta(1/2, 1/2)$ on the other hand, we got an estimated volume that was approximately 0 which was not close to the estimated value, while $Beta(2, 2)$ had an estimated value close to the true volume. Looking at the variance we can also see that we get a higher variance when sampling from a $Uniform[-1, 1]$ than when sampling with $Beta(2, 2)$.The 95% error bounds calculated (with the exception of the bounds calculated in 1.2 for the estimated volume calculated from sampling from $Beta(1/2, 1/2)$) captured the true volume.

2. Computing the Bessel Function

For $m \in \mathbb{N}$, the Bessel function of the first kind is:

$J_m(x) = \frac{1}{\pi} \int_0^\pi cos(mt - xsin(t))dt$ for $x \geq 0$.

Use Monte Carlo sampling within the interval $[0, \pi]$ to estimate the value of this function. Try this for $x = 1, 2, 4,$ and $m = 1, 2, 4$. Compare error bounds for your estimate and compare with the true value. Consider after rescaling the Beta distribution to be on $[0, \pi]$.

1. Sampling each $x_i \overset{\text{iid}}{\sim} Uniform[0, \pi]$

Similar to question 1, we perform for this part Monte Carlo Uniform sampling.

```r
# Bessel values
xm1 = besselJ(1, 1)
xm2 = besselJ(2, 2)
xm4 = besselJ(4, 4)
x1m2 = besselJ(1, 2)
x1m4 = besselJ(1, 4)
x2m1 = besselJ(2, 1)
x2m4 = besselJ(2, 4)
x4m2 = besselJ(4, 2)
x4m1 = besselJ(4, 1)


# Bessel function
phi.Bessel = function(x, m, t){

  f = (1/pi)*cos(m*t-x*sin(t))
  return(f)
}

# Monte Carlo Uniform Sampling
VMCU2 = function(num = 500, x, phi){
  mcval = runif(n = num, min = 0, max = pi)
  values = c()
  m = c(1, 2, 4)
  for (i in m){
    estp = phi(x = x, m = i, t = mcval)
    muh = mean(estp)*pi
    avar = (pi)^2*var(estp)/(num)
    values = c(values, muh, avar)
  }
  return(values)
}

# 95% error bounds
vol_CI = function(muh, varh, n, alpha){
  norm = c(muh-qnorm(1-alpha/2)*sqrt(varh),  muh+qnorm(1-alpha/2)*sqrt(varh))
  return(rbind(norm))
}
```

For $n = 10000$, we compute the Bessel function for the 9 possible combinations between $x = 1, 2, 4$ and $m = 1, 2, 4$:

```
n = 10000
x1 = VMCU2(num = n , x = 1, phi.Bessel)
x2 = VMCU2(num = n , x = 2, phi.Bessel)
x4 = VMCU2(num = n , x = 4, phi.Bessel)


univalTable = data.frame(
  Value = c(xm1, x1m2, x1m4, x2m1, xm2, x2m4,  x4m1, x4m2, xm4),
  Estimated_Value = c(x1[1], x1[3],  x1[5], x2[1], x2[3], x2[5], x4[1], x4[3], x4[5]),
  Variance = c( x1[2], x1[4],  x1[6], x2[2], x2[4], x2[6], x4[2], x4[4], x4[6]),
  Lower = c(vol_CI(x1[1], x1[2],n , 0.05)[1], vol_CI(x1[3], x1[4],n , 0.05 )[1], vol_CI(x1[5], x
1[6],n , 0.05 )[1], vol_CI(x2[1], x2[2],n , 0.05 )[1], vol_CI(x2[3], x2[4],n , 0.05 )[1], vol_CI
(x2[5], x2[6],n , 0.05 )[1], vol_CI(x4[1], x4[2],n , 0.05 )[1], vol_CI(x4[3], x4[4],n , 0.05 )
[1], vol_CI(x4[5], x4[6],n , 0.05 )[1]),

  Upper = c(vol_CI(x1[1], x1[2],n , 0.05 )[2], vol_CI(x1[3], x1[4],n , 0.05 )[2], vol_CI(x1[5],
x1[6],n , 0.05 )[2], vol_CI(x2[1], x2[2],n , 0.05 )[2], vol_CI(x2[3], x2[4],n , 0.05 )[2], vol_C
I(x2[5], x2[6],n , 0.05 )[2], vol_CI(x4[1], x4[2],n , 0.05 )[2], vol_CI(x4[3], x4[4],n , 0.05 )
[2], vol_CI(x4[5], x4[6],n , 0.05 )[2])
)

colnames(univalTable) = c("True Value  ", "Estimated Value  ", "Varaince  ", "Lower  ", "Upper
")
rownames(univalTable) = c("x = 1, m = 1 ", "x = 1, m = 2 ", "x = 1, m = 4 ", "x = 2, m = 1 ", "x
= 2, m = 2 ", "x = 2, m = 4 ", "x = 4, m = 1 ", "x = 4, m = 2 ", "x = 4, m = 4 ")
knitr::kable(univalTable)
```

	True Value	Estimated Value	Varaince	Lower	Upper
x = 1, m = 1	0.4400506	0.4420868	4.85e-05	0.4284308	0.4557427
x = 1, m = 2	0.1149035	0.1246317	5.04e-05	0.1107138	0.1385496
x = 1, m = 4	0.0024766	0.0054055	5.05e-05	-0.0085220	0.0193331
x = 2, m = 1	0.5767248	0.5856868	3.43e-05	0.5741999	0.5971736
x = 2, m = 2	0.3528340	0.3537275	5.14e-05	0.3396714	0.3677836
x = 2, m = 4	0.0339957	0.0486423	5.00e-05	0.0347773	0.0625072
x = 4, m = 1	-0.0660433	-0.0676788	4.41e-05	-0.0806873	-0.0546703
x = 4, m = 2	0.3641281	0.3602922	3.16e-05	0.3492705	0.3713138
x = 4, m = 4	0.2811291	0.2836417	5.33e-05	0.2693384	0.2979450

We can see from the output above that the variances are small, and most of the estimated values are approximately close to the values computed in R using besselJ() (labeled as True Value above). Both estimated and true values are within the 95% error bounds. We can also see that at $x = 1, m = 4$ the estimated value is slightly more different from the true value, but both values are within the 95% error bounds.

2. Sampling each $x_i \overset{\text{iid}}{\sim} Beta(1/2, 1/2)$

We first note that we have to re-scale the beta distribution to be on the interval $[0, pi]$. Similar to question 1.2 and 1.3, we re-scale in the following way: $y = x(\pi - 0) + 0 = x\pi$

```r
# Monte Carlo sampling with transformed Beta function
VMCB2 = function(n, x, nu, phi){
  mcval = pi*rbeta(n, nu, nu)
  values = c()
  m = c(1, 2, 4)
  for (i in m){
    gx = phi(x = x, m = i, t = mcval)
    w = dbeta(mcval/pi, shape1 =nu, shape2 = nu)

    muh = pi*mean(gx/w)
    avar = pi^(2)*var(gx/w)/n

    values = c(values, muh, avar)
  }

  return(values)
}
```

For $n = 10000$, we compute the Bessel function for the 9 possible combinations between $x = 1, 2, 4$ and $m = 1, 2, 4$:

```
n = 10000

bx1 = VMCB2(n, x =1, nu = 0.5, phi.Bessel)
bx2 = VMCB2(n, x =2, nu = 0.5, phi.Bessel)
bx4 = VMCB2(n, x =4, nu = 0.5, phi.Bessel)

bvalTable = data.frame(
  Value = c(xm1, x1m2, x1m4, x2m1, xm2, x2m4,  x4m1, x4m2, xm4),
  Estimated_Value = c(bx1[1], bx1[3],  bx1[5], bx2[1], bx2[3], bx2[5], bx4[1], bx4[3], bx4[5]),
  Variance = c( bx1[2], bx1[4],  bx1[6], bx2[2], bx2[4], bx2[6], bx4[2], bx4[4], bx4[6]),
  Lower = c(vol_CI(bx1[1], bx1[2],n , 0.05)[1], vol_CI(bx1[3], bx1[4],n , 0.05 )[1], vol_CI(bx1
[5], bx1[6],n , 0.05 )[1], vol_CI(bx2[1], bx2[2],n , 0.05 )[1], vol_CI(bx2[3], bx2[4],n , 0.05 )
[1], vol_CI(bx2[5], bx2[6],n , 0.05 )[1], vol_CI(bx4[1], bx4[2],n , 0.05 )[1], vol_CI(bx4[3], bx
4[4],n , 0.05 )[1], vol_CI(bx4[5], bx4[6],n , 0.05 )[1]),

  Upper = c(vol_CI(bx1[1], bx1[2],n , 0.05 )[2], vol_CI(bx1[3], bx1[4],n , 0.05 )[2], vol_CI(bx1
[5], bx1[6],n , 0.05 )[2], vol_CI(bx2[1], bx2[2],n , 0.05 )[2], vol_CI(bx2[3], bx2[4],n , 0.05 )
[2], vol_CI(bx2[5], bx2[6],n , 0.05 )[2], vol_CI(bx4[1], bx4[2],n , 0.05 )[2], vol_CI(bx4[3], bx
4[4],n , 0.05 )[2], vol_CI(bx4[5], bx4[6],n , 0.05 )[2])
)

colnames(bvalTable) = c("True Value  ", "Estimated Value  ", "Varaince  ", "Lower  ", "Upper  ")
rownames(bvalTable) = c("x = 1, m = 1 ", "x = 1, m = 2 ", "x = 1, m = 4 ", "x = 2, m = 1 ", "x =
2, m = 2 ", "x = 2, m = 4 ", "x = 4, m = 1 ", "x = 4, m = 2 ", "x = 4, m = 4 ")
knitr::kable(bvalTable)
```

	True Value	Estimated Value	Varaince	Lower	Upper
x = 1, m = 1	0.4400506	0.4381510	6.02e-05	0.4229462	0.4533559
x = 1, m = 2	0.1149035	0.1139594	5.68e-05	0.0991845	0.1287343
x = 1, m = 4	0.0024766	0.0040997	6.00e-05	-0.0110793	0.0192788
x = 2, m = 1	0.5767248	0.5654322	5.00e-05	0.5515797	0.5792847
x = 2, m = 2	0.3528340	0.3566556	6.22e-05	0.3411944	0.3721167
x = 2, m = 4	0.0339957	0.0403157	5.77e-05	0.0254285	0.0552028
x = 4, m = 1	-0.0660433	-0.0695839	5.16e-05	-0.0836579	-0.0555100
x = 4, m = 2	0.3641281	0.3581114	3.84e-05	0.3459729	0.3702499
x = 4, m = 4	0.2811291	0.2692316	6.37e-05	0.2535875	0.2848757

We can see from the output above that the variances are small yet slightly larger than the variances calculated through uniform sampling, and most of the estimated values are approximately close to the true values. Both estimated and true values are within the 95% error bounds. We can also see that just as in uniform sampling, at $x = 1, m = 4$ the estimated value is slightly higher compared to the true value, but both values are within the 95% error bounds.

3. Sampling each $x_i \overset{\text{iid}}{\sim} Beta(2,2)$

```
bx12 = VMCB2(n, x =1, nu = 2, phi.Bessel)
bx22 = VMCB2(n, x =2, nu = 2, phi.Bessel)
bx42 = VMCB2(n, x =4, nu = 2, phi.Bessel)

bvalTable = data.frame(
  Value = c(xm1, x1m2, x1m4, x2m1, xm2, x2m4,  x4m1, x4m2, xm4),
  Estimated_Value = c(bx12[1], bx12[3],  bx12[5], bx22[1], bx22[3], bx22[5], bx42[1], bx42[3], b
x42[5]),
  Variance = c( bx12[2], bx12[4],  bx12[6], bx22[2], bx22[4], bx22[6], bx42[2], bx42[4], bx42
[6]),
  Lower = c(vol_CI(bx12[1], bx12[2],n , 0.05)[1], vol_CI(bx12[3], bx12[4],n , 0.05 )[1], vol_CI
(bx12[5], bx12[6],n , 0.05 )[1], vol_CI(bx22[1], bx22[2],n , 0.05 )[1], vol_CI(bx22[3], bx22[4],
n , 0.05 )[1], vol_CI(bx22[5], bx22[6],n , 0.05 )[1], vol_CI(bx42[1], bx42[2],n , 0.05 )[1], vol
_CI(bx42[3], bx42[4],n , 0.05 )[1], vol_CI(bx42[5], bx42[6],n , 0.05 )[1]),

  Upper = c(vol_CI(bx12[1], bx12[2],n , 0.05 )[2], vol_CI(bx12[3], bx12[4],n , 0.05 )[2], vol_CI
(bx12[5], bx12[6],n , 0.05 )[2], vol_CI(bx22[1], bx22[2],n , 0.05 )[2], vol_CI(bx22[3], bx22[4],
n , 0.05 )[2], vol_CI(bx22[5], bx22[6],n , 0.05 )[2], vol_CI(bx42[1], bx42[2],n , 0.05 )[2], vol
_CI(bx42[3], bx42[4],n , 0.05 )[2], vol_CI(bx42[5], bx42[6],n , 0.05 )[2])
)

colnames(bvalTable) = c("True Value  ", "Estimated Value  ", "Varaince  ", "Lower  ", "Upper  ")
rownames(bvalTable) = c("x = 1, m = 1 ", "x = 1, m = 2 ", "x = 1, m = 4 ", "x = 2, m = 1 ", "x =
2, m = 2 ", "x = 2, m = 4 ", "x = 4, m = 1 ", "x = 4, m = 2 ", "x = 4, m = 4 ")
knitr::kable(bvalTable)
```

	True Value	Estimated Value	Varaince	Lower	Upper
x = 1, m = 1	0.4400506	0.4693920	0.0001525	0.4451864	0.4935977
x = 1, m = 2	0.1149035	0.1134912	0.0001551	0.0890785	0.1379040
x = 1, m = 4	0.0024766	-0.0021483	0.0001422	-0.0255222	0.0212257
x = 2, m = 1	0.5767248	0.5873655	0.0001189	0.5659939	0.6087372
x = 2, m = 2	0.3528340	0.3432721	0.0001396	0.3201137	0.3664305
x = 2, m = 4	0.0339957	0.0234635	0.0001300	0.0011195	0.0458075
x = 4, m = 1	-0.0660433	-0.0577095	0.0002028	-0.0856223	-0.0297967
x = 4, m = 2	0.3641281	0.3537234	0.0001941	0.3264145	0.3810323
x = 4, m = 4	0.2811291	0.2616365	0.0002173	0.2327413	0.2905317

We can see from the output above that the variances are larger than the variances calculated through uniform sampling and non-uniform sampling using a re-scaled $Beta(1/2, 1/2)$, and most of the estimated values are approximately close to the true values. We can see that for most cases, both estimated and true values are within the 95% error bounds except for $x = 1, m = 4$. We can also see that just as in uniform sampling, at $x = 1, m = 4$ the estimated value is negative.