PSTAT 115 Homework 4

Aaron Barel / Kevin Ayala

November 11, 2018

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------- tidyverse 1.2.1 --
## √ ggplot2 3.0.0     √ purrr   0.2.5
## √ tibble  1.4.2     √ dplyr   0.7.6
## √ tidyr   0.8.1     √ stringr 1.3.1
## √ readr   1.1.1     √ forcats 0.3.0
## -- Conflicts ------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

1.

(a)

```
set.seed(12345)

#True Expectation

true =  gamma(1.5)/0.5
true
```

```
## [1] 1.772454
```

```
dgal <- function(y){
  return(2 * (0.5^2) * y * exp(-1 * (0.5^2) * (y^2)))
}

q <- function(x){
  return(dexp(x,1))
}


# k = 10

estimate10 <- c()

for(n in 1:10){
  x10 <- rexp(10, 1) #sample
  w10 <- dgal(x10) / q(x10) #weight
  estimate10 <- c(estimate10, sum(x10 * w10)/ sum(w10)) #estimate
}

mean(estimate10) #sample expectation of estimate
```

```
## [1] 1.696885
```

```
# k = 100

estimate100 <- c()

for(n in 1:100){
  x100 <- rexp(100, 1)
  w100 <- dgal(x100) / q(x100)
  estimate100 <- c(estimate100, sum(x100 * w100)/ sum(w100))
}

mean(estimate100)
```

```
## [1] 1.775264
```

```
# k = 1000


estimate1000 <- c()

for(n in 1:1000){
  x1000 <- rexp(1000, 1)
  w1000 <- dgal(x1000) / q(x1000)
  estimate1000 <- c(estimate1000, sum(x1000 * w1000)/ sum(w1000))
}

mean(estimate1000)
```

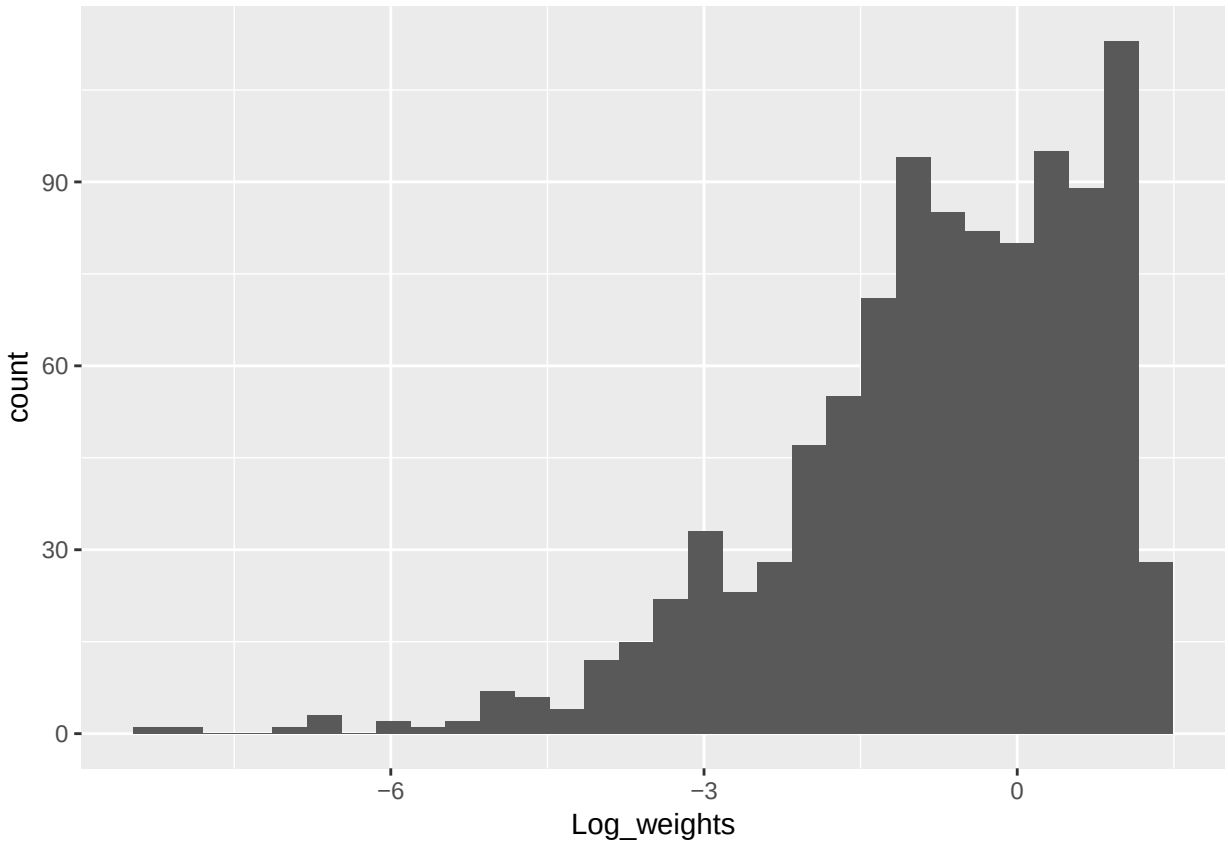
```
## [1] 1.771963
```

The true mean is 1.772454. For $k = 10$ our estimate was 1.696885, for $k = 100$ our estimate was 1.775264, and for $k = 1000$ our estimate was 1.771963. As we can see, both $k = 100$ and $k = 1000$ were very close estimates.

(b)

```
#histogram of log weights
ggplot(data.frame('Log_weights' = log(w1000)), aes(Log_weights)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



(c)

```
set.seed(562)

#True Expectation
true
```

```
## [1] 1.772454
```

```
q.1c <- function(x){
  return(dnorm(x, 1.5, 1))
}


# k = 10
estimate10.1c <- c()

for(n in 1:10){
  x10.1c <- rnorm(10, 1.5, 1)
  x10.1c <- x10.1c[x10.1c > 0]
  w10.1c <- dgal(x10.1c) / q.1c(x10.1c)
  estimate10.1c <- c(estimate10.1c, sum(x10.1c * w10.1c)/ sum(w10.1c))
}

mean(estimate10.1c)
```

```
## [1] 1.741471
```

```
# k = 100
estimate100.1c <- c()

for(n in 1:100){
  x100.1c <- rnorm(100, 1.5, 1)
  x100.1c <- x100.1c[x100.1c > 0]
  w100.1c <- dgal(x100.1c) / q.1c(x100.1c)
  estimate100.1c <- c(estimate100.1c, sum(x100.1c * w100.1c)/ sum(w100.1c))
}

mean(estimate100.1c)
```

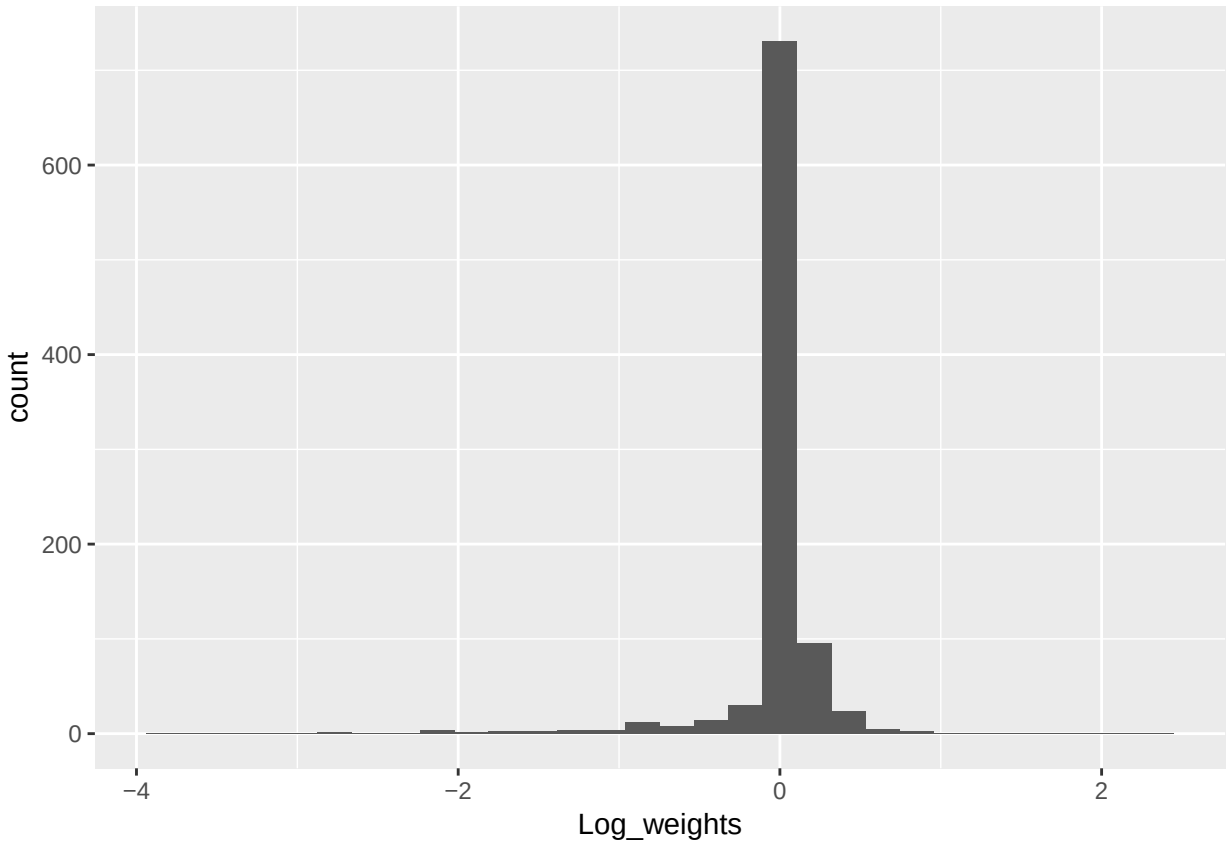
```
## [1] 1.758118
```

```
# k = 1000
estimate1000.1c <- c()

for(n in 1:1000){
  x1000.1c <- rnorm(1000, 1.5, 1)
  x1000.1c <- x1000.1c[x1000.1c > 0]
  w1000.1c <- dgal(x1000.1c) / q.1c(x1000.1c)
  estimate1000.1c <- c(estimate1000.1c, sum(x1000.1c * w1000.1c)/ sum(w1000.1c))
}

mean(estimate1000.1c)
```

```
## [1] 1.772368
```

```
#histogram of log weights
#hist(log(w1000.1c))
ggplot(data.frame('Log_weights' = log(w1000.1c)), aes(Log_weights)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



While the estimates are close, and error can be due to the random seed, the histogram for the normal proposal is much better than the exponential proposal because it has much less variance. Therefore, we will conclude the normal proposal is better.

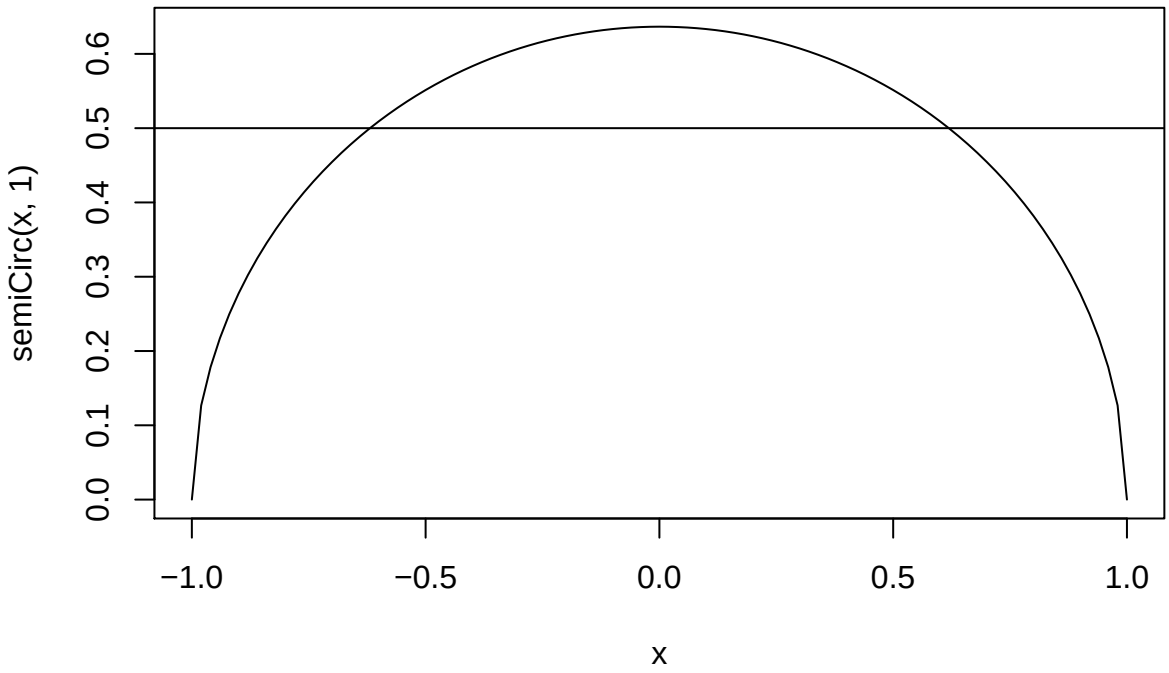
2.

(a)

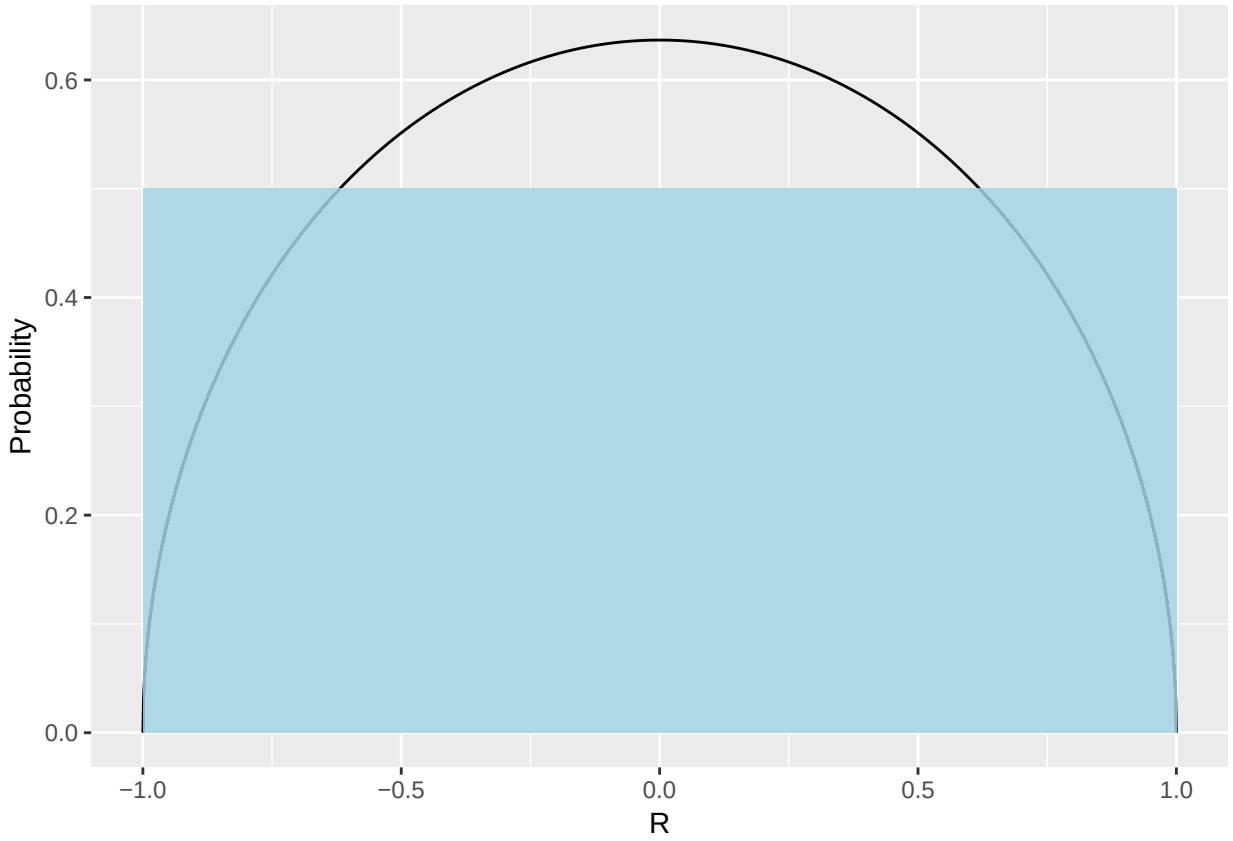
We can have the proposal distribution, $g(x|R)$, be a uniform distribution $\frac{1}{2R}$ since this is the length of the support of the distribution of $p(x|R)$. To find M, we compute the $max(\frac{p(x|R)}{g(x|R)}) = max(\frac{4}{\pi R}\sqrt{R^2 - x^2})$. Since this is a semi-circle centered at 0, we know the maximum is at $x = 0$. Therefore $M = \frac{4}{\pi}$, and since M is finite we know $Mg(x|R)$ "envelopes" $p(x|R$. We can see $Mg(x|R) = \frac{2}{\pi R} \geq p(x|R) = \frac{2}{\pi R^2}\sqrt{R^2 - x^2}$, $\forall x \in [-R, R]$.

(b)

```
semiCirc <- function(x, R){
  return(2 / (pi * R^2) * sqrt(R^2 - x^2))
}
curve(semiCirc(x, 1), from = -1, to = 1) #semi circle
abline(h = 0.5) #proposal density
```

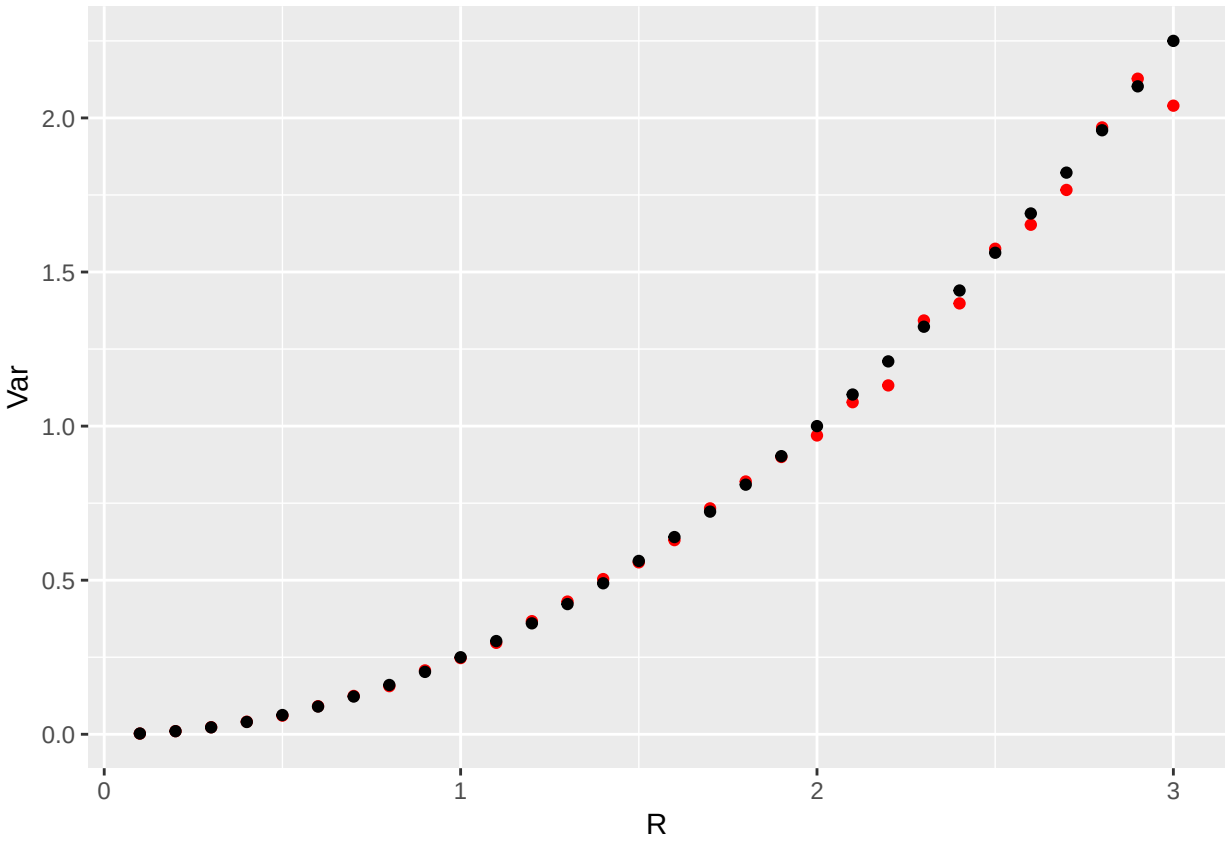



```
#ggplot looks a bit nicer
ggplot(data.frame('R' = seq(-1,1, by = 0.001), 'Semi_Circle' = semiCirc(seq(-1,1, by = 0.001), 1))) + g
```



(c)

```
set.seed(69)

R = seq(0.1, 3, 0.1)
#max
M = 4/pi
#true variance
true_var <- R^2 / 4
#variances for each R
var_eachR <- c()
for (r in R){
  #draw sample from g(x|R)
  g_samples <- runif(1000, -r, r)
  #accepted sample vector
  samples.accept <- c()
  #pdf vector with samples as inputs
  p.2c <- semiCirc(g_samples, r)
  #Accept sample as draw w prob p(x|R)/Mg(x|R)
  prob.2c <- p.2c / (M * dunif(g_samples, -r, r))
  #probabilities
  randomProb <- runif(1000,0,1)

  for(j in 1:1000){
    if(randomProb[j] <= prob.2c[j]){
      samples.accept <- c(samples.accept, g_samples[j])
    }
  }

  var_eachR <- c(var_eachR, var(samples.accept))
}

plot_var <- data.frame('R' = R, 'Sample_Variance' = var_eachR, "True_Variance" = true_var)


ggplot(plot_var) + geom_point(aes(x = R, y = Sample_Variance), color = 'red') + geom_point(aes(x = R, y
```



As the plot shows, the sample variance is extremely close to the true variance with error being random by set.seed().

3.

(a)

```
set.seed(1997)

library(HDInterval)

p.3 <- function(x){
  return(abs(sin(x)))
}

accepted_samples <- c()
#Proposal Uniform = 1/2pi, M = 1

samples.3a <- runif(1000, 0, 2*pi)
rprob <- runif(1000, 0, 1)
p.3a <- p.3(samples.3a)
prob.3a <- p.3a / dunif(samples.3a, 0, 2*pi)

for(i in 1:1000){
  if(rprob[i] <= prob.3a[i]){
```

```
    accepted_samples <- c(accepted_samples, samples.3a[i])
  }
}

mass = 0.5

ql <- quantile(accepted_samples, (1-mass) / 2)
ql
```

```
##      25%
## 1.496489
```

```
qh <- quantile(accepted_samples, mass + (1-mass) / 2)
qh
```

```
##      75%
## 4.734789
```

```
length_quant <- as.numeric(qh - ql)
length_quant
```

```
## [1] 3.2383
```

```
hd <- hdi(density(accepted_samples), credMass = mass, allowSplit = TRUE)


hd[1,1];hd[1,2]
```

```
##     begin
## 0.5876595
```

```
##      end
## 2.143551
```

```
hd[2,1];hd[2,2]
```

```
##    begin
## 4.143982
```

```
##      end
## 5.528896
```

```
length_hd1 <- as.numeric(hd[1,2] - hd[1,1])
length_hd1
```

```
## [1] 1.555891
```

```
length_hd2 <- as.numeric(hd[2,2] - hd[2,1])
length_hd2
```

```
## [1] 1.384914
```

```
total_length_hdi <- length_hd1 + length_hd2
total_length_hdi
```

```
## [1] 2.940805
```

The quantile region is $(1.496489, 4.734789)$ with a length of 3.2383. The HPD region is $(0.5876595, 2.143551) \bigcup (4.143982, 5.528$ so the absolute lower bound is 0.5876595 and the absolute upper bound is 5.528896, this is because sine is periodic and therefore bimodal. The HPD total length is 2.940805. The HPD interval should always be

smaller or equal to the quantile interval for the same mass. It is smaller than the HPD interval despite having more extreme bounds because it is a union of two intervals.

(b)

```
curve(p.3(x), 0, 2*pi)
segments(ql, 0.01, qh, 0.01, col = 'blue')
abline(v = ql, col = 'blue', lty = 2)
abline(v = qh, col = 'blue', lty = 2)
segments(hd[1,1], 0, hd[1,2], 0, col = 'red')
abline(v = hd[1,1], col = 'red', lty = 2)
abline(v = hd[1,2], col = 'red', lty = 2)
segments(hd[2,1], 0, hd[2,2], 0, col = 'red')
abline(v = hd[2,1], col = 'red', lty = 2)
abline(v = hd[2,2], col = 'red', lty = 2)
```