MSiA 400 Lab Assignment 3

Matt Ko: mjk3551

Problem 1

I rolled a 6-sided die 100 times and observed the following results:

1	2	3	4	5	6
18	11	9	25	18	19

Problem 1a

What is the maximum likelihood estimate for the dice roll probabilities $\boldsymbol{\theta} = (\theta_1, \cdots, \theta_6)$?

$$\theta = \frac{n_i}{\sum_{i=1}^{N} n_i}$$

Problem 1b

Assume the prior $\boldsymbol{\theta} \sim \text{Diri}(\mathbf{1})$. I.e., assume that prior over roll probabilities are uniform Dirichlet with prior $\boldsymbol{\alpha} = 1$ (the Dirichlet distribution is a multivariate generalization of the beta distribution, with probability distribution function $p(\boldsymbol{\theta}|\boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{i=1}^{K} \theta_i^{\alpha_i - 1}$). I.e., use a prior assumption that the die is fair and using six artificial rolls (one on each face) to incorporate this prior.

What is the posterior log-likelihood of the above rolls (in terms of $\boldsymbol{\theta}$)? What is the maximum a posteriori estimate for $\boldsymbol{\theta}$?

Log-likelihood

$$\sum_{i=1}^{N} (n_i + \alpha_i - 1)(\ln(\theta_i)) = (18 * ln(\theta_1)) + (11 * ln(\theta_2)) + (9 * ln(\theta_3)) + (25 * ln(\theta_4)) + (18 * ln(\theta_5)) + (19 * ln(\theta_6))$$

MAP

$$\theta = \frac{n_i + \alpha_i - 1}{\sum_{i=1}^{N} n_i + \alpha_i - 1}$$

### For alpha = 1

$$\theta = \frac{n_i}{\sum_{i=1}^{N} n_i}$$

Problem 2

I have one fair die (all rolls have 1/6 probability) and one weighted die, with roll probabilities as follows:

1	2	3	4	5	6
2/13	2/13	1/13	4/13	2/13	2/13

Model dice rolls as a Hidden Markov model, with the hidden state corresponding to whether or not the die is fair. Assume a prior of 50/50 that the initial roll is from a fair die and a probability of 25% (after each roll) that I swap the dice. The following rolls are observed:

4, 4, 5, 2, 2, 4, 6, 6, 1, 4, 1, 1, 3, 5, 5, 2, 5, 4, 2, 1

Problem 2a

What is the posterior distribution of the initial hidden state, i.e. what is the probability that the first roll is made with a fair die? Note: this depends solely on the second hidden state and the first roll. What is the posterior distribution for the final hidden state? What is the posterior distribution for the rolls in between (this depends solely on the hidden states before and after it and the given roll)?

First roll fair die: $0.5 * \frac{1}{6} = \frac{1}{12}$

Posterior First Roll

$\pi_1 = Fair$

$$P(\pi_0 = Fair|\pi_1 = Fair, x_0 = 4) = \frac{P(\pi_1 = Fair, x_0 = 4|x_0 = Fair)P(\pi_0 = Fair)}{P(\pi_1 = Fair, x_0 = 4)}$$

$$= \frac{P(\pi_1 = Fair, x_0 = 4|\pi_0 = Fair)P(\pi_0 = Fair)}{P(\pi_1 = Fair, x_0 = 4)P(\pi_0 = Fair) + P(\pi_1 = Fair, x_0 = 4|\pi_0 = Weight)P(\pi_0 = Weight)}$$

$$= \frac{P(\pi_1 = Fair|\pi_0 = Fair)P(x_0 = 4|\pi_0 = Fair)P(\pi_0 = Fair)}{P(\pi_1 = Fair|\pi_0 = Fair)P(x_0 = 4|\pi_0 = Fair)P(\pi_0 = Fair) + P(\pi_1 = Fair|\pi_0 = Weight)P(x_0 = 4|\pi_0 = Weight)P(\pi_0 = W}$$

$$= \frac{0.75 * \frac{1}{6} * \frac{1}{2}}{0.75 * \frac{1}{6} * \frac{1}{2} + 0.25 * \frac{4}{3} * \frac{1}{2}}$$

$$= 0.619$$

$\pi_1 = Weight$

$$P(\pi_0 = Fair|\pi_1 = Weight, x_0 = 4) = \frac{P(\pi_1 = Weight, x_0 = 4|x_0 = Fair)P(\pi_0 = Fair)}{P(\pi_1 = Weight, x_0 = 4)}$$

$$= \frac{P(\pi_1 = Weight|\pi_0 = Fair)P(x_0 = 4|\pi_0 = Fair)P(\pi_0 = Fair)}{P(\pi_1 = Weight|\pi_0 = Fair)P(x_0 = 4|\pi_0 = Fair)P(\pi_0 = Fair) + P(\pi_1 = Weight|\pi_0 = Weight)P(x_0 = 4|\pi_0 = Weight)P(\pi_0}$$

$$= \frac{0.25 * \frac{1}{6} * \frac{1}{2}}{0.25 * \frac{1}{6} * \frac{1}{2} + 0.75 * \frac{4}{3} * \frac{1}{2}}$$
$$= 0.153$$

Posterity Final State

$\pi_1 = Fair$

$$P(\pi_{19} = Fair|\pi_{18} = Fair, x_{19} = 1) = \frac{P(x_{19} = 1|\pi_{19} = Fair)P(\pi_{18} = Fair|\pi_{19} = Fair)}{P(\pi_{18} = Fair, x_{19} = 1)}$$

$$= \frac{P(x_{19} = 1|\pi_{19} = Fair)P(\pi_{18} = Fair|\pi_{19} = Fair)}{P(\pi_{18} = Fair, x_{19} = 1|\pi_{19} = Fair) + P(\pi_{18} = Fair, x_{19} = 1|\pi_{19} = Weight)}$$

$$= \frac{\frac{1}{6} * 0.75}{0.75 * \frac{1}{6} + 0.25 * \frac{2}{13}}$$

$$= 0.765$$

$\pi_1 = Weight$

$$P(\pi_{19} = Fair|\pi_{18} = Weight, x_{19} = 1) = \frac{P(x_{19} = 1|\pi_{19} = Fair)P(\pi_{18} = Weight|\pi_{19} = Fair)}{P(\pi_{18} = Weight, x_{19} = 1)}$$

$$= \frac{P(x_{19} = 1|\pi_{19} = Fair)P(\pi_{18} = Weight|\pi_{19} = Fair)}{P(\pi_{18} = Weight, x_{19} = 1|\pi_{19} = Fair) + P(\pi_{18} = Weight, x_{19} = 1|\pi_{19} = Weight)}$$

$$= \frac{\frac{1}{6} * 0.25}{0.25 * \frac{1}{6} + 0.75 * \frac{2}{13}}$$

$$= 0.265$$

In Between States

$$P(\pi_x = A|\pi_{x-1} = B, x_x = y, \pi_{x+1} = C) = \frac{P(x_x = y|\pi_x = A)P(\pi_{x-1} = B|\pi_x = A)P(\pi_{x+1} = C|\pi_x = A)}{P(\pi_{x-1} = B, x_x = y, \pi_{x-1} = C)}$$

$$P(\pi_x = A|\pi_{x-1} = B, x_x = y, \pi_{x+1} = C) = \frac{P(x_x = y|\pi_x = A)P(\pi_{x-1} = B|\pi_x = A)P(\pi_{x+1} = C|\pi_x = A)}{P(\pi_{x-1} = B, x_x = y, \pi_{x+1} = C|\pi_x = A) + P(\pi_{x-1} = B, x_x = y, \pi_{x+1} = C|\pi_x = \quad \sim A}$$

Problem 2b

Estimate the probability that each roll is from a fair die, using a Gibbs sampler. Plot this probability over the sequence of rolls.

```r
firstlast = function(p0,p1,x0){
  if(p1=='W'){
    if(p0=='F'){
      num = 0.25*(1/6)
      denom = 0.25*(1/6)+0.75*(wprob[x0])
      return(num/denom)
    }
    else{
      num = 0.75*(wprob[x0])
      denom = 0.75*(wprob[x0])+0.25*(1/6)
      return(1-(num/denom))
    }

  }
  else{
    if(p0=='F'){
      num = 0.75*(1/6)
      denom = 0.75*(1/6)+0.25*(wprob[x0])
      return(num/denom)
    }
    else{
      num = 0.25*(wprob[x0])
      denom = 0.75*(wprob[x0])+0.25*(1/6)
      return(1-(num/denom))
    }
  }

}
```

```
between = function(px,pb,x,pa){

  if(pb=='W'){
    if(pa=='F'){
      if(px=='F'){
        num = (1/6)*0.25*0.75
        denom = 0.25*(1/6)*0.75+0.75*(wprob[x])*0.25
        return(num/denom)
      }
      else{
        num = (wprob[x])*0.75*0.25
        denom = 0.75*(wprob[x])*0.25+0.25*(1/6)*0.75
        return(1-(num/denom))
      }

    }
    else{
      if(px=='F'){
        num = (1/6)*0.25*0.25
        denom = 0.25*(1/6)*0.25+0.75*(wprob[x])*0.75
        return(num/denom)
      }
      else{
        num = (wprob[x])*0.75*0.75
        denom = 0.75*(wprob[x])*0.75+0.25*(1/6)*0.25
        return(1-(num/denom))

      }
    }
  }
  else{
    if(pa=='F'){
      if(px=='F'){
        num = (1/6)*0.75*0.75
        denom = 0.75*(1/6)*0.75+0.25*(wprob[x])*0.25
        return(num/denom)
      }
      else{
        num = (wprob[x])*0.25*0.25
        denom = 0.25*(wprob[x])*0.25+0.75*(1/6)*0.75
        return(1-(num/denom))

      }
    }
    else{
      if(px=='F'){
        num = (1/6)*0.75*0.25
        denom = 0.75*(1/6)*0.25+0.25*(wprob[x])*0.75
        return(num/denom)
      }
      else{
        num = (wprob[x])*0.25*0.75
        denom = 0.25*(wprob[x])*0.75+0.75*(1/6)*0.25
        return(1-(num/denom))
      }
    }
  }

}
```

```
#set.seed(123)
counts = c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
rolls = c(4, 4, 5, 2, 2, 4, 6, 6, 1, 4, 1, 1, 3, 5, 5, 2, 5, 4, 2, 1)
states = c('F', 'F', 'F', 'F', 'F', 'F', 'F', 'F', 'F', 'F', 'F', 'F', 'F', 'F', 'F', 'F', 'F', 'F', 'F',
wprob = c(2/13, 2/13, 1/13, 4/13, 2/13, 2/13)
for(i in 1:15000){

  for(j in 1:20){

    if(j==1){
      p = firstlast(states[1],states[2],rolls[1])
    }
    else if(j==20){
      p = firstlast(states[20],states[19],rolls[20])
    }
    else{
      p = between(states[j],states[j-1],rolls[j],states[j+1])
    }
    r = runif(1)
    #cat(p,' ',j,' ')
    if(p>r){
      states[j]='F'
      counts[j] = counts[j] + 1
    }
    else{
      states[j]='W'
    }
  }

}
```

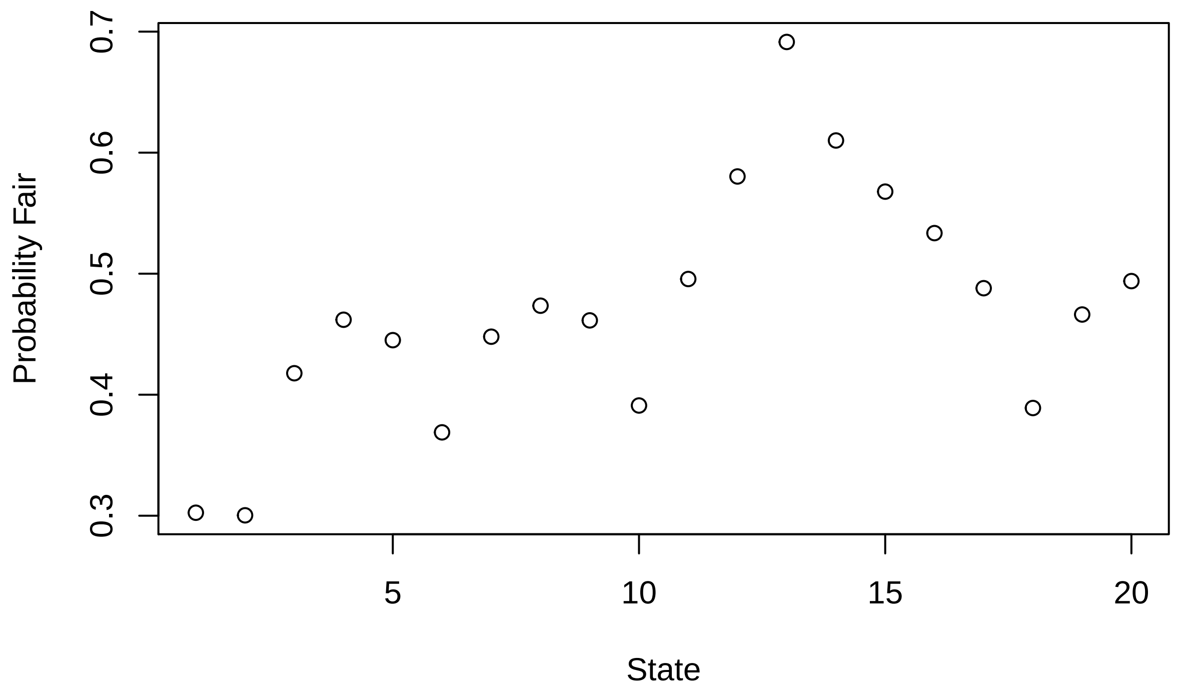
```
counts/15000
```

```
##  [1] 0.3024667 0.3003333 0.4177333 0.4619333 0.4450667 0.3688667 0.4479333
##  [8] 0.4735333 0.4614000 0.3910667 0.4956000 0.5803333 0.6914667 0.6100667
## [15] 0.5678000 0.5335333 0.4880000 0.3890000 0.4662667 0.4938667
```

```
seq(1,20)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
plot(seq(1,20),counts/15000,xlab = 'State',ylab = 'Probability Fair')
```



Problem 3

You will further analyze the `gradAdmit.csv` dataset (from Lab 4 and Assignment 2). As a reminder, this dataset contains a list of students (rows), along with whether or not they were admitted to graduate school (`admit`), their GRE score (`gre`), their GPA (`gpa`), and the prestige of their undergraduate university (`rank`). You do not need to repeat the parameter tuning from Assignment 2.

```r
admits <- read.csv("gradAdmit.csv")
set.seed(314159)
n = nrow(admits) # number of samples
# hold out 20% for testing
sample = sample.int(n = n, size = floor(.2*n), replace = F)
train = admits[-sample,]; test = admits[sample,]
```

Problem 3a

Compute the class balance for both the training set (80% from Assignment 2) and test set (20%). For each dataset, what percentage of students were admitted?

```r
#Train set
table(train$admit)/nrow(train)
```

```
##
##        0        1
## 0.684375 0.315625
```

31.5 percent of students were admitted

```r
#Test set
table(test$admit)/nrow(test)
```

```
##
##     0     1
## 0.675 0.325
```

32.5 percent of students were admitted

Problem 3b

Using your optimal parameters from Assignment 2, Problem 1c, and the model trained on the full training set (if you did this improperly before, redo it), compute the precision, recall, and specificity of the test dataset. Hint: the `confusionMatrix` function may be helpful.

```
library(e1071)
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
smodel <- svm(factor(admit)~.,kernel='polynomial',cost=1,gamma=1,coef0=0,degree=5, data=train)
testpredict = predict(smodel,newdata=test)
finaltest = confusionMatrix(factor(test$admit), testpredict)
finaltest$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.7500000            0.6250000            0.8888889
##       Neg Pred Value            Precision               Recall
##            0.3846154            0.8888889            0.7500000
##                   F1           Prevalence       Detection Rate
##            0.8135593            0.8000000            0.6000000
## Detection Prevalence    Balanced Accuracy
##            0.6750000            0.6875000
```

Problem 3c

Based on your answer to Problem 3a, what percentage of minority over-sampling would create the most even class balance? Generate that many artificial training samples using the SMOTe algorithm (you may use the `SMOTE` function). Combine the original training dataset with the generated dataset and confirm the class balance is as desired.

```
library(DMwR)
```

```
## Loading required package: grid

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

The percentage would be ~116 = (0.684375 - 0.315625) / 0.315625 percent of minority over-sampling.

```
train$admit = as.factor(train$admit)
s <- SMOTE(admit~.,data=train,perc.over = 116, perc.under = 0)
#train <- rbind(train,s)
table(s$admit)
```

```
##
##   0   1
##   0 202
```

```
maj <- train[which(train$admit==0),]
nTrain <- rbind(s,maj)
table(nTrain$admit)
```

```
##
##   0   1
## 219 202
```

Problem 3d

Retrain your model on the combined training dataset (using the same parameters). Compute the precision, recall, and specificity of the test dataset. Note: the test dataset should not be augmented. How do they differ from Problem 3b?

```
smodel <- svm(admit~.,kernel='polynomial',cost=1,gamma=1,coef0=0,degree=5, data=nTrain)
testpredict = predict(smodel,newdata=test)
finaltest = confusionMatrix(factor(test$admit), testpredict)
finaltest$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.7037037            0.3396226            0.3518519
##       Neg Pred Value            Precision               Recall
##            0.6923077            0.3518519            0.7037037
##                   F1           Prevalence       Detection Rate
##            0.4691358            0.3375000            0.2375000
## Detection Prevalence    Balanced Accuracy
##            0.6750000            0.5216632
```

The precision and specificity decreased, while the recall increased for the model on the combined dataset.

Problem 4

Use importance sampling and the Monte Carlo integration method to estimate the integral $\int\limits_{10\pi}^{\infty} e^{-x}\sin x dx$. Use $p(x) = e^{-x}$ and $g(x) = \begin{cases} \sin x, & x \geq 10\pi \\ 0, & x < 10\pi \end{cases}$. Note: this problem is similar to Assignment 1, Problem 2.

Problem 4a

What is the probability of drawing a sample $x \geq 10\pi$ from the exponential distribution (with $\lambda = 1$), i.e. $p(x \geq 10\pi)$?

```
pexp(10*pi,lower.tail = FALSE)
```

```
## [1] 0.00000000000002271101
```

Problem 4b

What is the exact solution to the integral, i.e., the result obtained via calculus? You may use the result given in Assignment 1, Problem 2: $\int\limits_{0}^{\infty} e^{-\lambda x}\sin x dx = \frac{1}{1+\lambda^2}$.

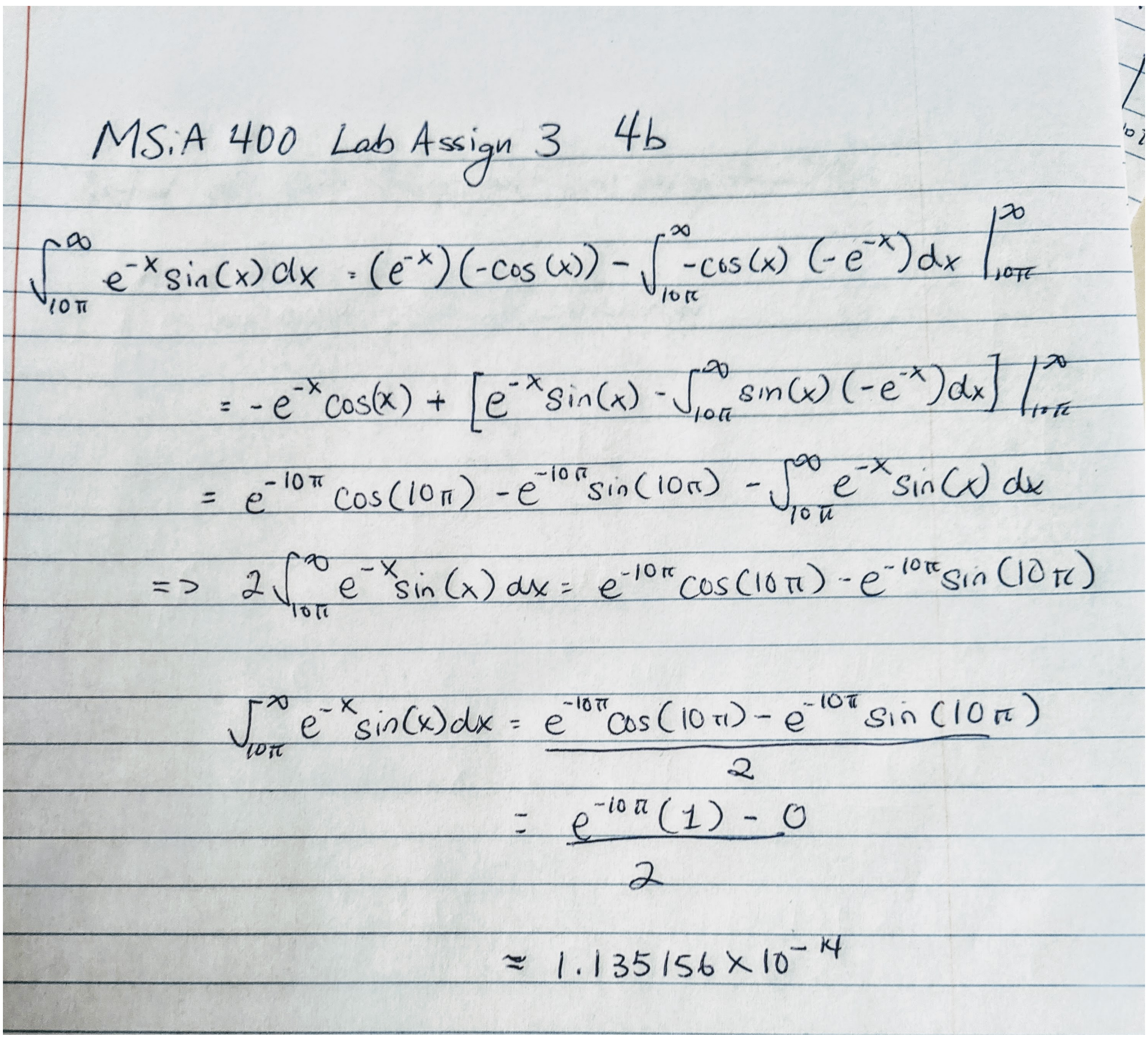
```
integrand = function(x) {exp(-1*x)*sin(x)}
integrate(integrand, lower = 10*pi, upper = Inf)
```

```
## 0.00000000000001135156 with absolute error < 0.0000000000000071
```

Problem 4c

Pick a biasing distribution that should work well for this problem. Your goal is to minimize the variance. Explain your choice. Note: choosing $p^*(x) > p(x)$ when $g^2(x)p(x)$ is large and $p^*(x) < p(x)$ when $g^2(x)p(x)$ is small reduces the variance.

Figure 1: '4b'



MSiA 400 Lab Assign 3 4b

$$\int_{10\pi}^{\infty} e^{-x}\sin(x)\,dx = (e^{-x})(-\cos(x)) - \int_{10\pi}^{\infty} -\cos(x)\,(-e^{-x})\,dx \,\Big|_{10\pi}^{\infty}$$

$$= -e^{-x}\cos(x) + \left[e^{-x}\sin(x) - \int_{10\pi}^{\infty} \sin(x)\,(-e^{-x})\,dx\right] \Big|_{10\pi}^{\infty}$$

$$= e^{-10\pi}\cos(10\pi) - e^{-10\pi}\sin(10\pi) - \int_{10\pi}^{\infty} e^{-x}\sin(x)\,dx$$

$$\Rightarrow \quad 2\int_{10\pi}^{\infty} e^{-x}\sin(x)\,dx = e^{-10\pi}\cos(10\pi) - e^{-10\pi}\sin(10\pi)$$

$$\int_{10\pi}^{\infty} e^{-x}\sin(x)\,dx = \frac{e^{-10\pi}\cos(10\pi) - e^{-10\pi}\sin(10\pi)}{2}$$

$$= \frac{e^{-10\pi}(1) - 0}{2}$$

$$\approx 1.135156 \times 10^{-14}$$

Choosing a shifted exponential distribution. This distribution allows for probabilities to fit the criteria above versus a different distribution that only changed the rates that will not result in much of a bias.

Problem 4d

Numerically estimate the integral using the importance sampling method with the biasing distribution from Problem 4c and number of samples $n = 10^6$.

```r
set.seed(12345)
tot = 0
for (i in 1:1000000){
  x_star = rexp(1, rate = 1) + 10*pi
  if (x_star>=10*pi){
    g = sin(x_star)
  }else{
    g = 0
  }
  r = pexp(x_star, lower.tail = FALSE)/pexp(x_star-10*pi, lower.tail = FALSE, rate = 1)
  prod = r*g
  tot = tot + prod


}
tot/1000000
```

```
## [1] 0.00000000000001135953
```