Homework 9

Louis Bensard

April 30, 2018

Problem 1:

(a)

Let $\lambda > 0$, we want to show that there exists a $x > 0$ in $\mathbb{R}$ such that $f(x) = g(x)$.

$$
\begin{aligned}
f(x) = g(x) &\Leftrightarrow \frac{1}{\sqrt{2\pi}} \cdot e^{\frac{-x^2}{2}} = \frac{\lambda e^{\lambda|x|}}{2} = \frac{\lambda e^{\lambda x}}{2} \, (since \, x > 0) \\
&\Leftrightarrow \frac{1}{2}x^2 - \lambda x - ln(\frac{2}{\lambda\sqrt{2\pi}}) = 0 \quad (1) \\
&\Rightarrow \Delta = \lambda^2 - 2ln(\lambda) + ln(\frac{2}{\pi}) \geq 0
\end{aligned}
$$

Now let's see if $\Delta \geq 0$ for all *lambda*, ie (1) has a real solution:

$$
\begin{aligned}
\frac{d\Delta}{d\lambda} = 2\lambda - \frac{2}{\lambda} = 0 \Rightarrow \lambda = 1 \, (since \, \lambda > 0) \\
\Rightarrow \frac{d^2\Delta}{d\lambda^2} = 2 + \frac{2}{\lambda^2} > 0
\end{aligned}
$$

So $\lambda = 1$ is a minimum. Moreover, $\Delta(\lambda = 1) = 1 + ln(\frac{2}{\pi}) > 0$. Therefore, for all $\lambda > 0$, $\Delta > 0$ and thus (1) has a real solution.

As a result, for any $\lambda > 0$ the double exponential density and the standard normal density intersect at a value of $x > 0$. (One of them is $\lambda + \sqrt{\Delta} > 0$).

The two density are symmetric around the y-axis, therefore the above results is true for $x < 0$ as well by symmetry.

(b)

Let's plot f and g so see what we are dealing with:

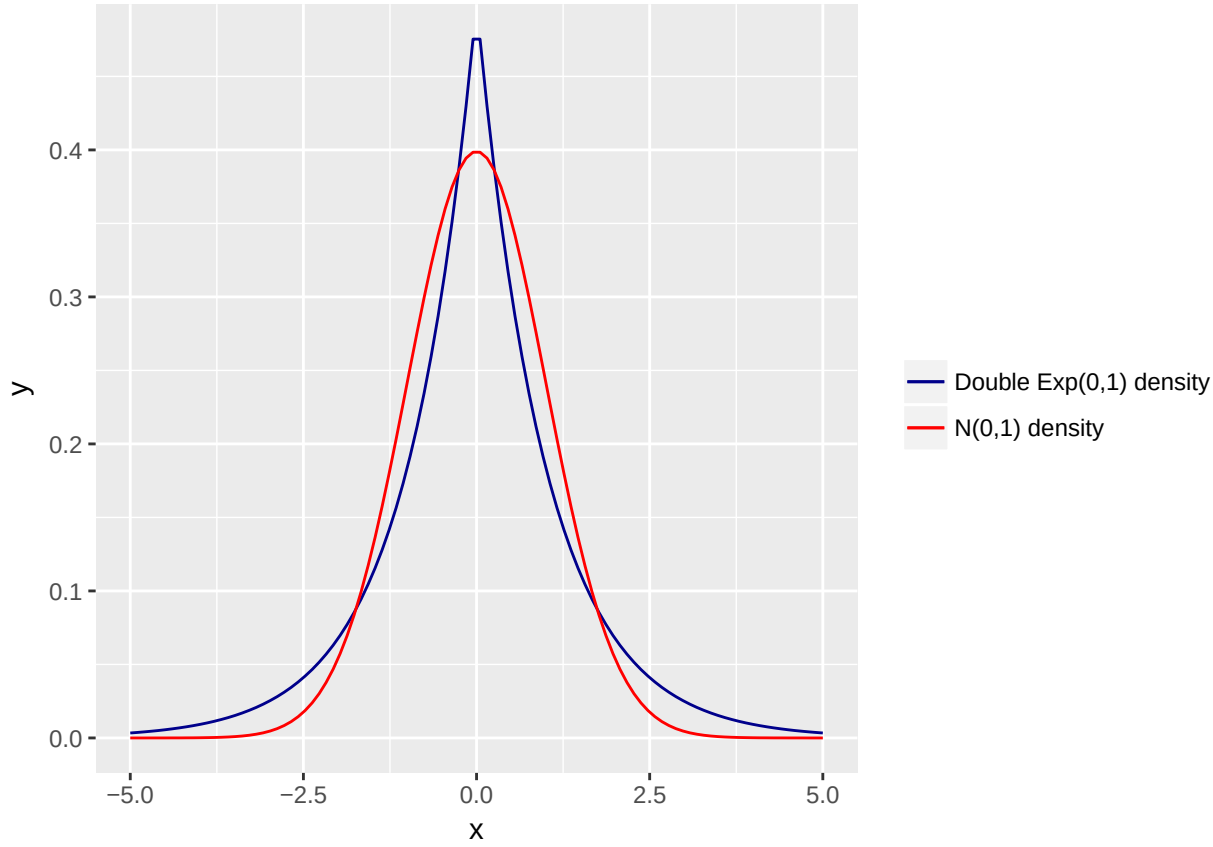
```
library(ggplot2)
library(smoothmest)
```

```
## Warning: package 'smoothmest' was built under R version 3.4.4
```

```
## Loading required package: MASS
```

```
x = seq(-5,5, length=100)

f <- function(x) return(dnorm(x,0,1))

g <- function(x) return(ddoublex(x,0,1))
```

```
df = data.frame(x, y1=f(x), y2=g(x))
ggplot(df,aes(x=x))+
    geom_line(aes(y=y2, colour="Double Exp(0,1) density"))+
    geom_line(aes(y=y1, colour="N(0,1) density"))+
    labs(y="y")+
    scale_colour_manual("", values=c('darkblue','red'))
```



We can see they intersect each other as proven in (a), now let's find a optimal c to make g envelope f.

Let $\lambda = 1$, we want to find $c > 1$ such that for all $x \in \mathbb{R}$:

$$c \cdot g(x) \geq f(x) \Leftrightarrow c \cdot \frac{e^{-|x|}}{2} \geq \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}} \quad (2)$$

By symmetry, it is sufficient to work with only $x > 0$, therefore, since we want the smallest c that satisfies (2), rearranging the equation gives us:

$$c \geq \sqrt{\frac{2}{\pi}} \cdot e^{\frac{-x^2}{2}+x}$$

Now we need the x^* that minimizes the equation:

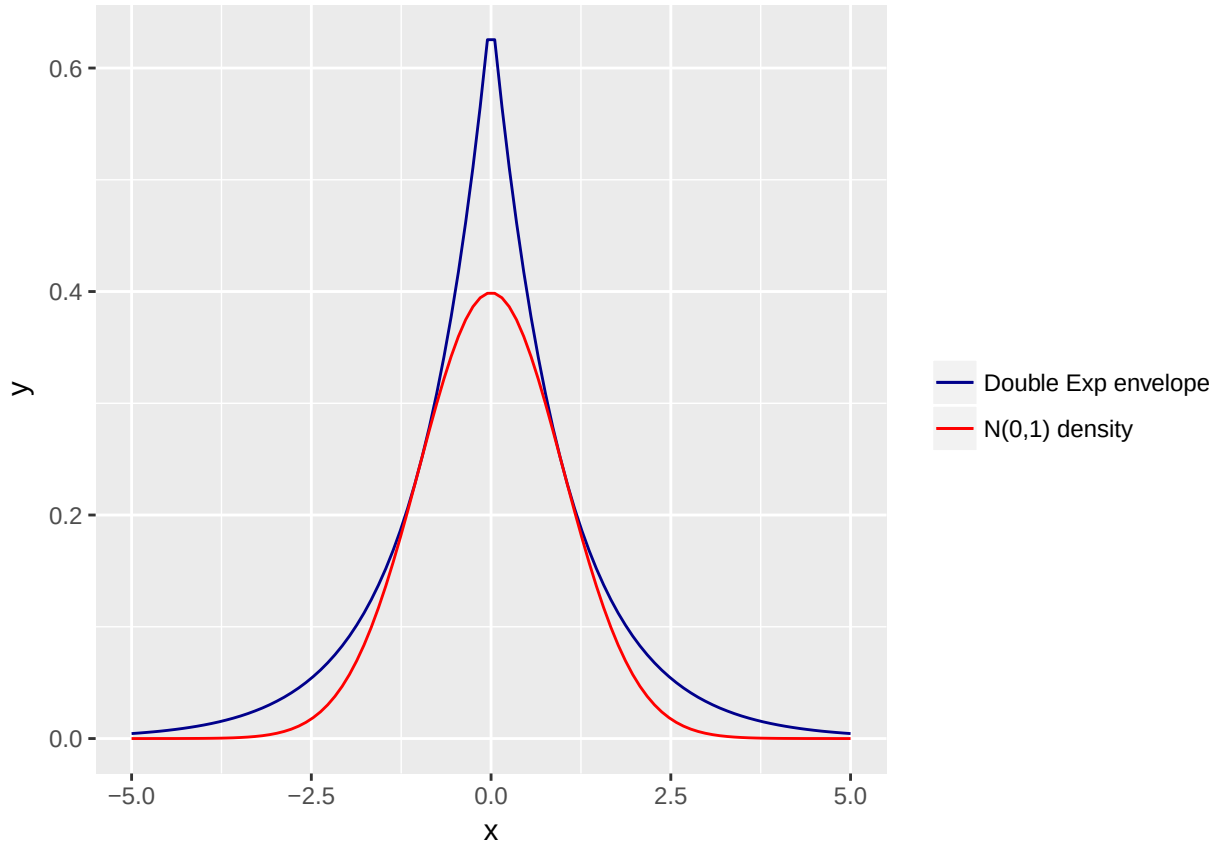
$$log(c) = \frac{1}{2}log(\frac{2}{\pi}) - \frac{x^2}{2} + x$$
$$\Rightarrow \frac{dlog(c)}{dx} = -x + 1 \Rightarrow x^* = 1$$

As a result, for $\lambda = 1$ we get the following optimal c:

$$c^* = \sqrt{\frac{2}{\pi}} \cdot e^{\frac{1}{2}} \simeq 1.3155$$

Let's plot the resulting envelope :

```
c_star = sqrt(2/pi)*exp(1/2)

df = data.frame(x, y1=f(x), y2=c_star*g(x))
ggplot(df,aes(x=x))+
    geom_line(aes(y=y2, colour="Double Exp envelope"))+
    geom_line(aes(y=y1, colour="N(0,1) density"))+
    labs(y="y")+
    scale_colour_manual("", values=c('darkblue','red'))
```

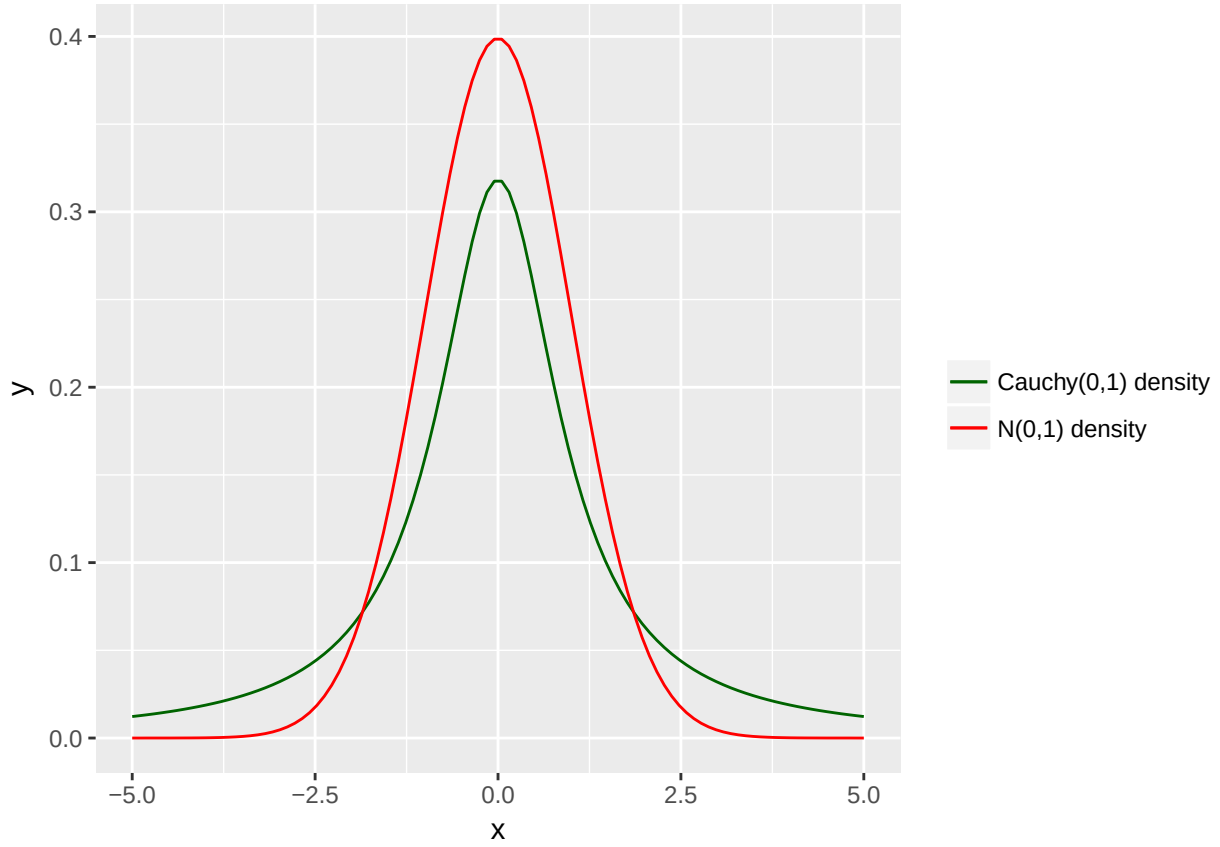


$c \cdot g(x)$ envelope $f(x)$ as close a possible for all x , this is what we wanted.

(c)

a. Cauchy envelope

```
g1 <- function(x) return(dcauchy(x, 0,1))

df = data.frame(x, y1=f(x), y2=g1(x))
ggplot(df,aes(x=x))+
    geom_line(aes(y=y2, colour="Cauchy(0,1) density"))+
    geom_line(aes(y=y1, colour="N(0,1) density"))+
    labs(y="y")+
    scale_colour_manual("", values=c('darkgreen','red'))
```



Let $g_1(x)$ be the Cauchy(0,1) density. First, the same way we did in part (b), let's find an optimal $c_1 > 1$ such that $c_1 \cdot g_1(x) \geq f(x)$. Rearranging the equation, we have:

$$c_1 \geq \sqrt{\frac{\pi}{2}} \cdot e^{\frac{-x^2}{2}} \cdot (1 + x^2)$$

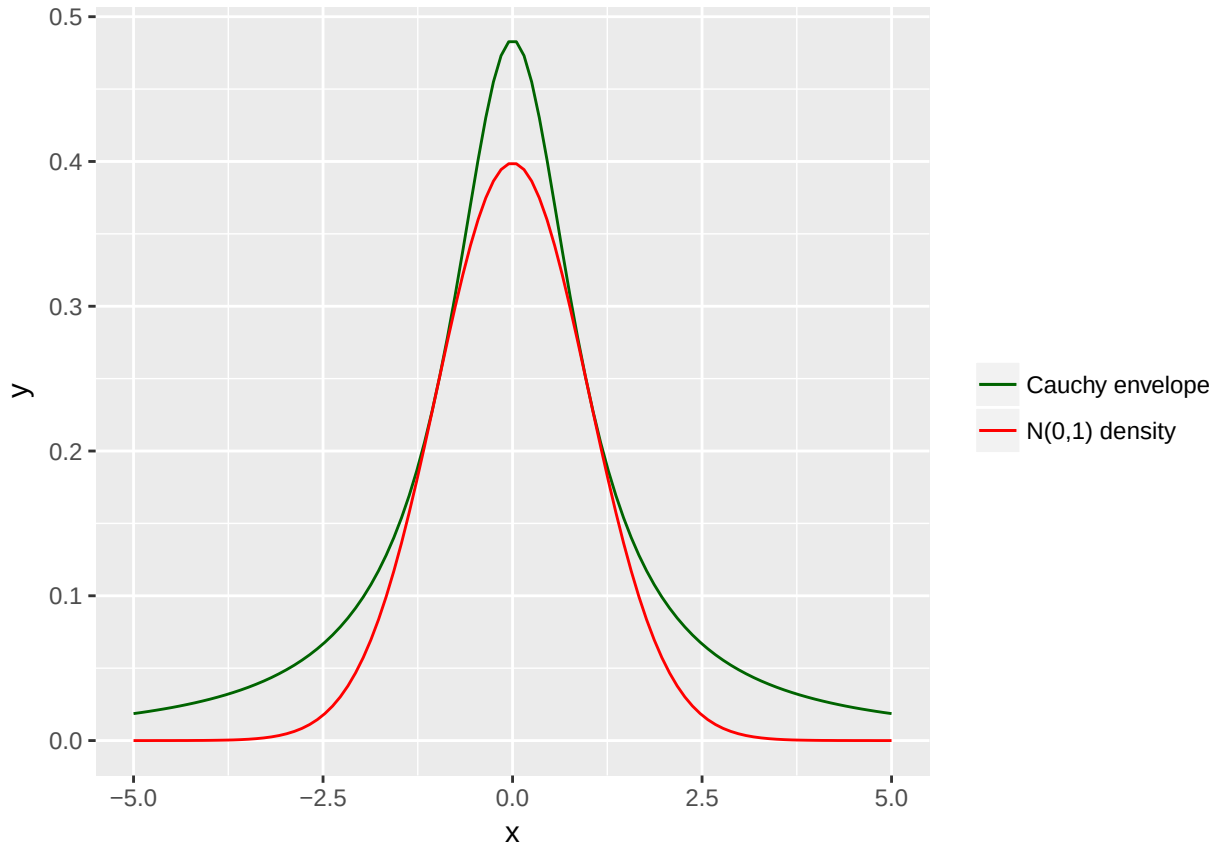
Now we need the x^* that minimizes the equation:

$$log(c_1) = \frac{1}{2}log(\frac{\pi}{2}) - \frac{x^2}{2} + log(1+x^2)$$
$$\Rightarrow \frac{dlog(c_1)}{dx} = -x + \frac{2x}{1+x^2} \Rightarrow x^* = 1$$

As a result, for a Cauchy(0,1) we get the following optimal c:

$$c^* = \sqrt{2\pi} \cdot e^{-\frac{1}{2}} \simeq 1.5204$$

```
c1 = sqrt(pi/2)*exp(-1/2)*2

#envolope
e1 <- function(x) return(c1*g1(x))

df = data.frame(x, y1=f(x), y2=e1(x))
ggplot(df,aes(x=x))+
    geom_line(aes(y=y2, colour="Cauchy envelope"))+
    geom_line(aes(y=y1, colour="N(0,1) density"))+
    labs(y="y")+
    scale_colour_manual("", values=c('darkgreen','red'))
```



Now, let's apply the Accept/Reject algorithm to generate X from the standard normal distribution, by generating $U \sim uniform(0,1)$ and using our optimal Cauchy envelope e_1.

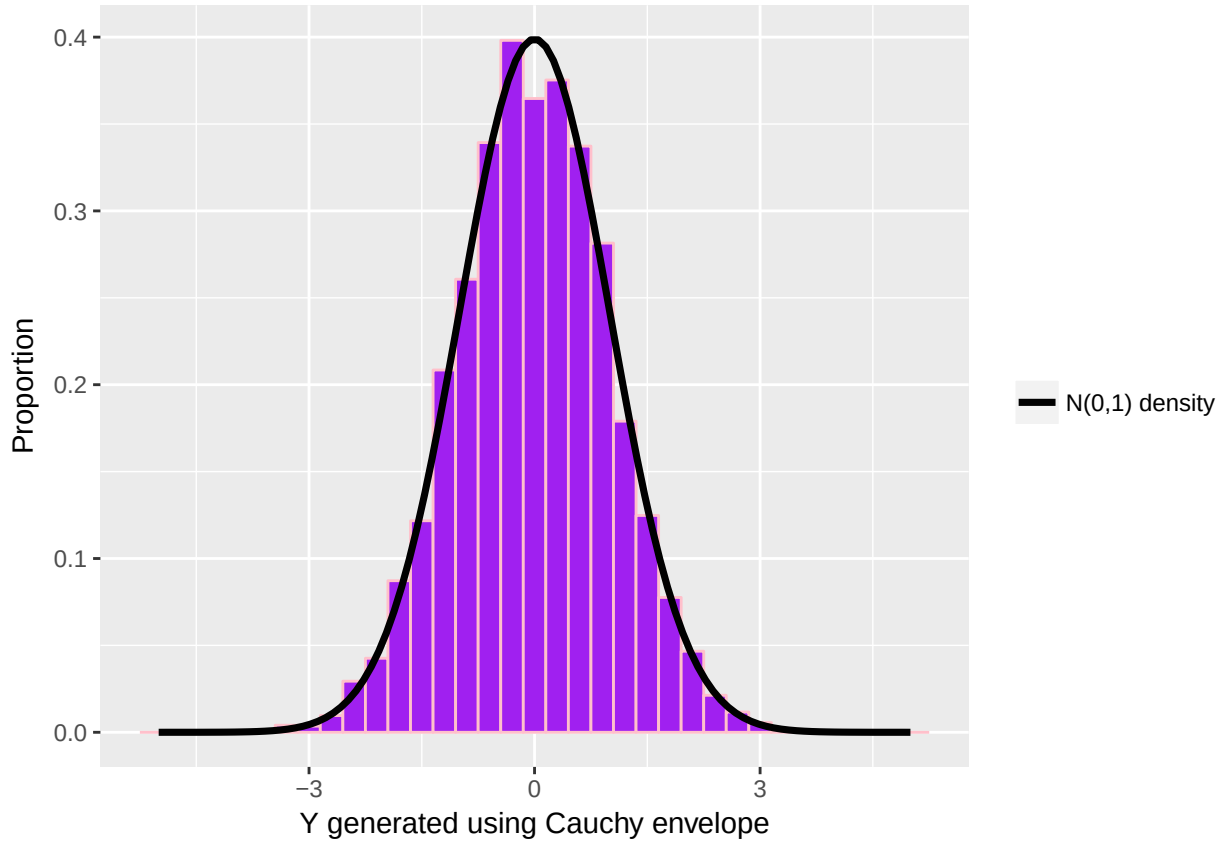
```r
#Accept/rejet algorithm

n= 10000
Y = rcauchy(n,0,1)

U  = runif(n,0,1)

Y = Y[U<=f(Y)/e1(Y)]

print(mean(Y)); print(sd(Y)) #mean=0 and sd = 1, looks good
```

```
## [1] -0.01344765
```

```
## [1] 1.005866
```

The mean ($\simeq 0$) and standard error ($\simeq 1$) look good since we want to generate from N(0,1). Let's plot an histogram.

```r
df = data.frame(Y)
df1 = data.frame(x, y1=f(x))
ggplot(data=df, aes(x = Y)) +
    geom_histogram(aes(y=..density..), fill="purple", color="pink", binwidth=.3)+
    geom_line(data=df1, aes(x=x, y=y1, colour="N(0,1) density"), lwd=1.25)+
    labs(x="Y generated using Cauchy envelope", y='Proportion')+
    scale_colour_manual("", values='black')
```



The histogram looks Normal(0,1), so we are confident to say that we successfuly generated $X's$ from N(0,1).

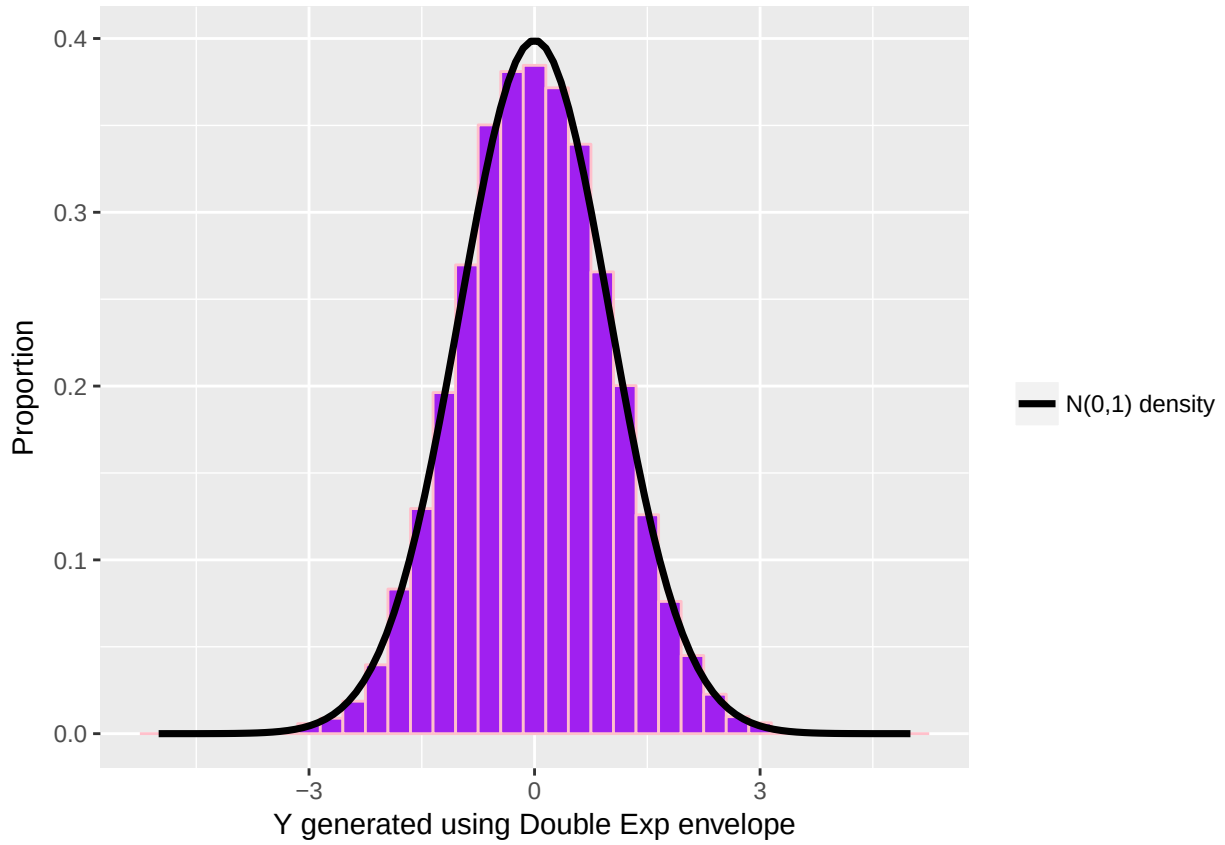
b. Double Exponential envelope

```
g2 = g #from part a)

c2 = c_star #from part a)

#envolope
e2 <- function(x) return(c2*g2(x))

#Accept/rejet algorithm
n= 10000
Y = rdoublex(n,0,1)

U  = runif(n,0,1)

Y = Y[U<=f(Y)/e2(Y)]

print(mean(Y)); print(sd(Y)) #mean~0 and sd ~ 1, looks good
```

```
## [1] -0.0009444556
```

```
## [1] 0.991605
```

```
df = data.frame(Y)
df1 = data.frame(x, y1=f(x))
ggplot(data=df, aes(x = Y)) +
    geom_histogram(aes(y=..density..), fill="purple",color="pink", binwidth=.3)+
    geom_line(data=df1, aes(x=x, y=y1, colour="N(0,1) density"), lwd=1.25)+
    labs(x="Y generated using Double Exp envelope", y='Proportion')+
    scale_colour_manual("", values='black')
```



The histogram looks Normal(0,1) as well, so we are confident to say that we successfuly generated $X's$ from N(0,1).

c.

I think using an envelope of Double Exponential is better than an envelope of Cauchy. Indeed, we know that the number of $X \sim N(0,1)$ generated is about $\frac{n}{c1}$ for Cauchy and about $\frac{n}{c2}$ for Double Exponential. But since $c2 \simeq 1.31 < 1.52 \simeq c1$, then we generate more $X's$ using the Double Exp than using Cauchy, for the same number n of $uniform(0,1)$ and g_1 or g_2 generations. That's because the Double Exp envelope fits The normal density better so we reject less of those generated values.

Problem 2: SIR Algorithm

Step 1:

First, let's generate m values of θ, λ_1 and λ_2 from their respective prior.

```
data = read.table('C:/Users/Louis/Documents/UPMC/M1/Spring 2018/MATH 534/Homework 9/coal_data.txt',
                  header=TRUE)

m = 50000

theta = sample(1:111, m, replace=T)
```

```
a1 = rgamma(m,shape=10, rate=10)
lambda1 = rgamma(m, shape=3, rate=a1)

a2 = rgamma(m,shape=10,rate=10)
lambda2 = rgamma(m,shape=3,rate=a2)
```

Step 2:

Now, let's compute the Likelihood L and use it to compute the weights. The given distribution of the data is $X_1, ..., X_\theta \sim Poisson(\lambda_1)$ and $X_{\theta+1}, ..., X_{112} \sim Poisson(\lambda_2)$. Thus we have:

$$\begin{aligned} L(\theta, \lambda_1, \lambda_2|x) &= \prod_{i=1}^{\theta} \frac{e^{-\lambda_1}\lambda_1^{x_i}}{x_i!} \prod_{i=\theta+1}^{112} \frac{e^{-\lambda_2}\lambda_2^{x_i}}{x_i!} \\ &\propto \prod_{i=1}^{\theta} e^{-\lambda_1}\lambda_1^{x_i} \prod_{i=\theta+1}^{112} e^{-\lambda_2}\lambda_2^{x_i} \end{aligned}$$

Note that to compute the weights, we only need the Likelihood up to a constant of proportionality. For computation purposes, let's compute the log-likelihood $l = log(L)$ and we will simply take the exponential of it before computing the weights:

$$l(\theta, \lambda_1, \lambda_2|x) = -\lambda_1\theta - \lambda_2(112-\theta) + log(\lambda_1)\sum_{i=1}^{\theta} x_i + log(\lambda_2)\sum_{i=\theta+1}^{112} x_i$$

Thus, for each set of parameter, with $i = 1, ..., m$, we have the following standardized weights:

$$w(\theta^{(i)}, \lambda_1^{(i)}, \lambda_2^{(i)}) = \frac{L(\theta^{(i)}, \lambda_1^{(i)}, \lambda_2^{(i)})}{\sum_{i=1}^{m} L(\theta^{(i)}, \lambda_1^{(i)}, \lambda_2^{(i)})}$$

```
#log-likelihood
l <- function(data, theta, lambda1, lambda2){

    sum1_vect = c(); sum2_vect = c()

    for(i in 1:m){

        x1 = data[1:theta[i], 2]
        x2 = data[(theta[i]+1):112, 2]
        sum1_vect = c(sum1_vect, sum(x1))
        sum2_vect = c(sum2_vect, sum(x2))
    }

    return(-lambda1*theta - lambda2*(112-theta) + log(lambda1)*sum1_vect + log(lambda2)*sum2_vect)
}

log_L = l(data, theta, lambda1, lambda2)
L = exp(log_L) #likelihood
weights = L/sum(L)
```

Step 3:

Now, we are going the resample n values of θ, λ_1 and λ_2 with replacement from the initial samples, but with drawing probability equal to the corresponding weights.

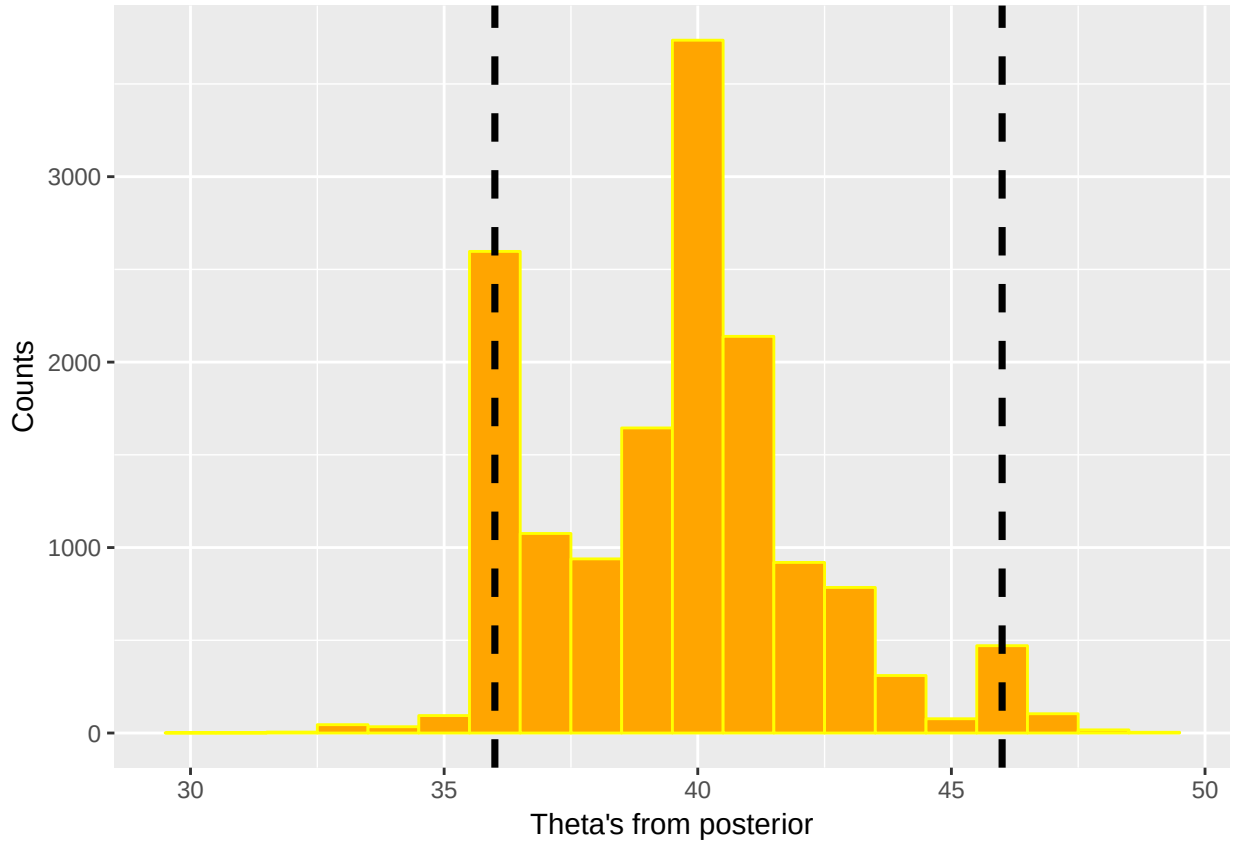
```
n = 15000
resample_indices = sample(1:m, n, replace=TRUE, prob=weights)

theta_new = theta[resample_indices]
lambda1_new = lambda1[resample_indices]
lambda2_new = lambda2[resample_indices]
```

Here is histograms and credible intervals for resampled θ, λ_1 and λ_2:

```
#95% credible interval and histogram of theta
L = sort(theta_new)[n*0.025]
U = sort(theta_new)[n*0.975]
cat("\nCI=[",L,",", U,"]")
```

```
##
## CI=[ 36 , 46 ]
```

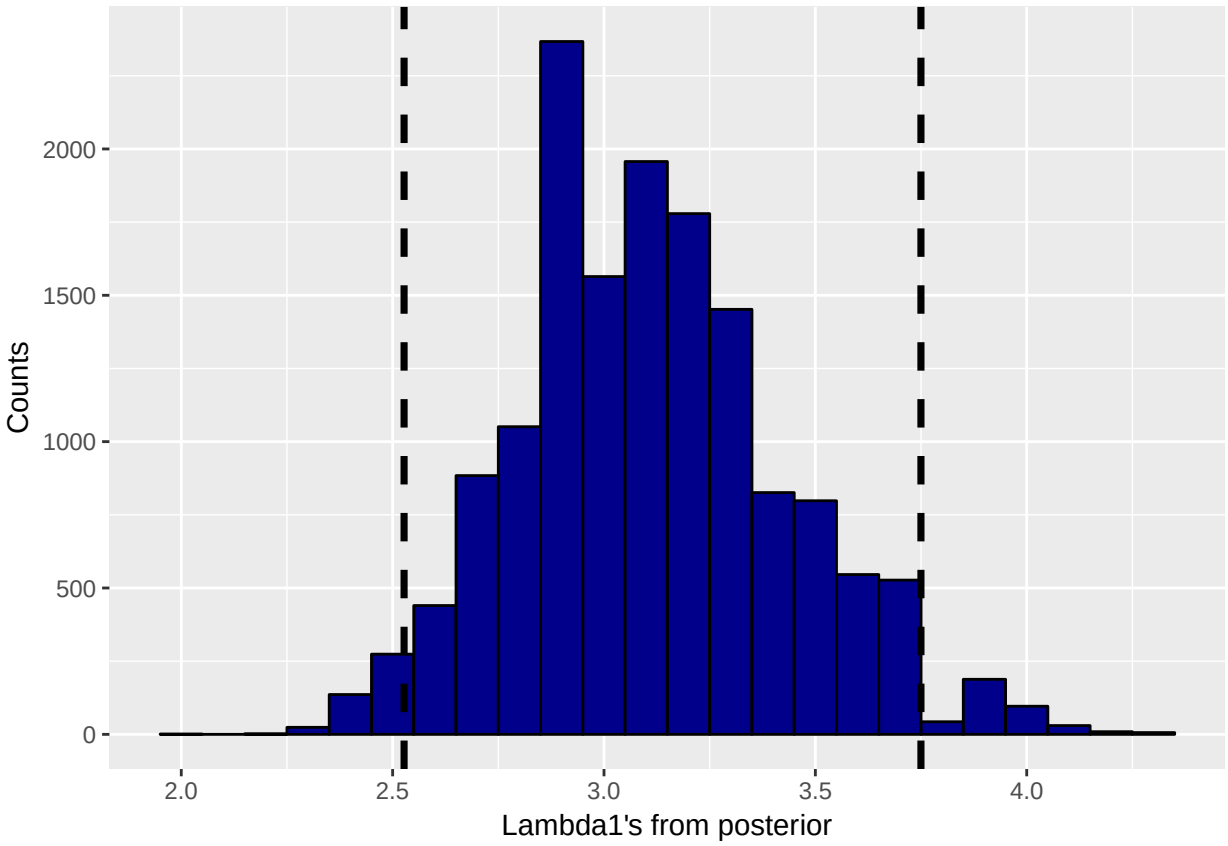
```
df = data.frame(x = theta_new)
ggplot(df, aes(x = x)) +
    geom_histogram(fill="orange", color="yellow", binwidth=1)+
    geom_vline(xintercept=L, lwd=1.25, lty=2)+
    geom_vline(xintercept=U, lwd=1.25, lty=2)+
    labs(x="Theta's from posterior", y='Counts')
```



```
#95% credible interval and histogram of lambda1
L = sort(lambda1_new)[n*0.025]
U = sort(lambda1_new)[n*0.975]
cat("\nCI=[",L,",", U,"]")
```

```
##
## CI=[ 2.527055 , 3.749741 ]
```

```
df = data.frame(x = lambda1_new)
ggplot(df, aes(x = x)) +
    geom_histogram(fill="darkblue", color="black", binwidth=.1)+
    geom_vline(xintercept=L, lwd=1.25, lty=2)+
    geom_vline(xintercept=U, lwd=1.25, lty=2)+
    labs(x="Lambda1's from posterior", y='Counts')
```

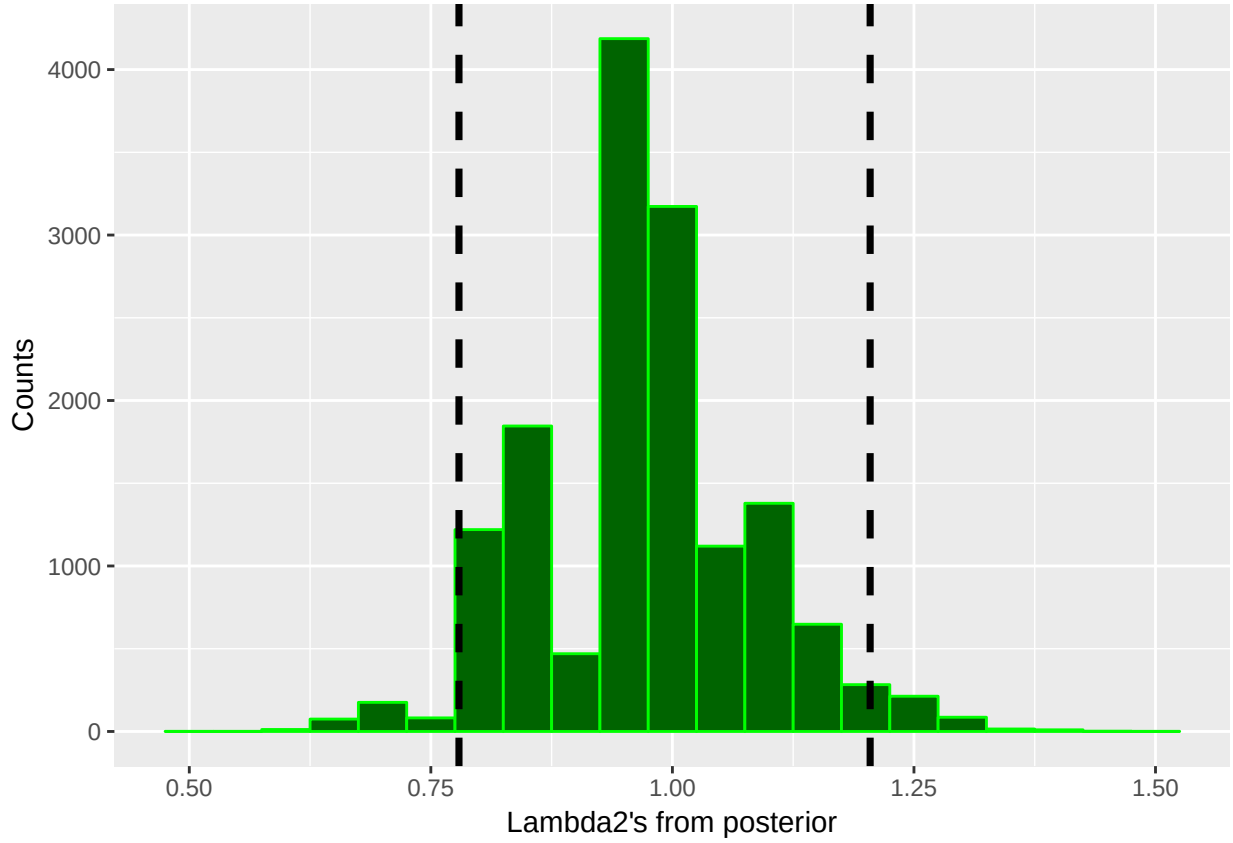



```
#95% credible interval and histogram of lambda2
L = sort(lambda2_new)[n*0.025]
U = sort(lambda2_new)[n*0.975]
cat("\nCI=[",L,",", U,"]")
```

```
##
## CI=[ 0.7791169 , 1.204668 ]
```

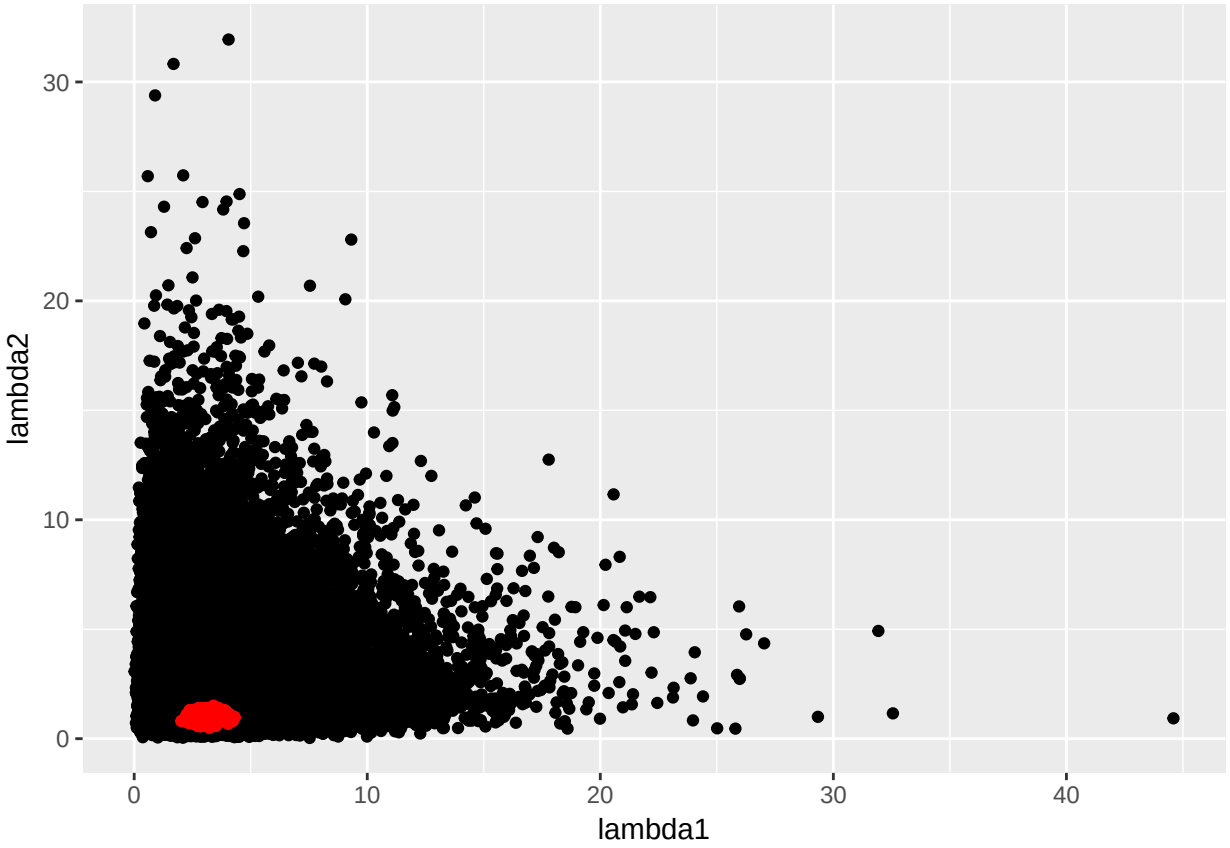
```
df = data.frame(x = lambda2_new)
ggplot(df, aes(x = x)) +
    geom_histogram(fill="darkgreen", color="green", binwidth=.05)+
    geom_vline(xintercept=L, lwd=1.25, lty=2)+
```

```
    geom_vline(xintercept=U, lwd=1.25, lty=2)+
    labs(x="Lambda2's from posterior", y='Counts')
```



Here is a plot of the initial sample of λ_1 vs the initial sample of λ_2 (black), overlaid by the resample of λ_1 vs the resample of λ_2 (red).

```
df = data.frame(lambda1, lambda2)
df1 = data.frame(lambda1_new, lambda2_new)
ggplot(df, aes(x=lambda1, y=lambda2))+
    geom_point()+
    geom_point(data=df1, aes(x=lambda1_new, y=lambda2_new), color="red")
```



We can clearly see that the resampling is weigthed on the small red aread on the plot. That is because those red points are now drawn from the posterior and not the prior.

I picked an initial sample size $m = 50000$ to represent fully the population. I wanted to pick a resample size n such that any larger resample size would not make much difference over the results, so I picked $n = 15000$.

Here is, for each parameter, the number of unique points in each resample and its highest observed frequency:

```
unique_theta = length(unique(theta_new))
cat("Number of unique points in the theta resample = ",unique_theta,"\n")
```

```
## Number of unique points in the theta resample =  20
```

```
t = table(theta_new); print(t[which.max(t)])
```

```
##   40
## 3736
```

```
unique_lambda1 = length(unique(lambda1_new))
cat("\nNumber of unique points in the lambda1 resample = ",unique_lambda1,"\n")
```

```
##
## Number of unique points in the lambda1 resample =  272
```

```
t = table(lambda1_new); print(t[which.max(t)])
```

```
## 3.13341888745892
##              992
```

```r
unique_lambda2 = length(unique(lambda2_new))
cat("\nNumber of unique points in the lambda2 resample = ",unique_lambda2,"\n")
```

```
##
## Number of unique points in the lambda2 resample =  272
```

```r
t = table(lambda2_new); print(t[which.max(t)])
```

```
## 0.971858582912059
##               992
```