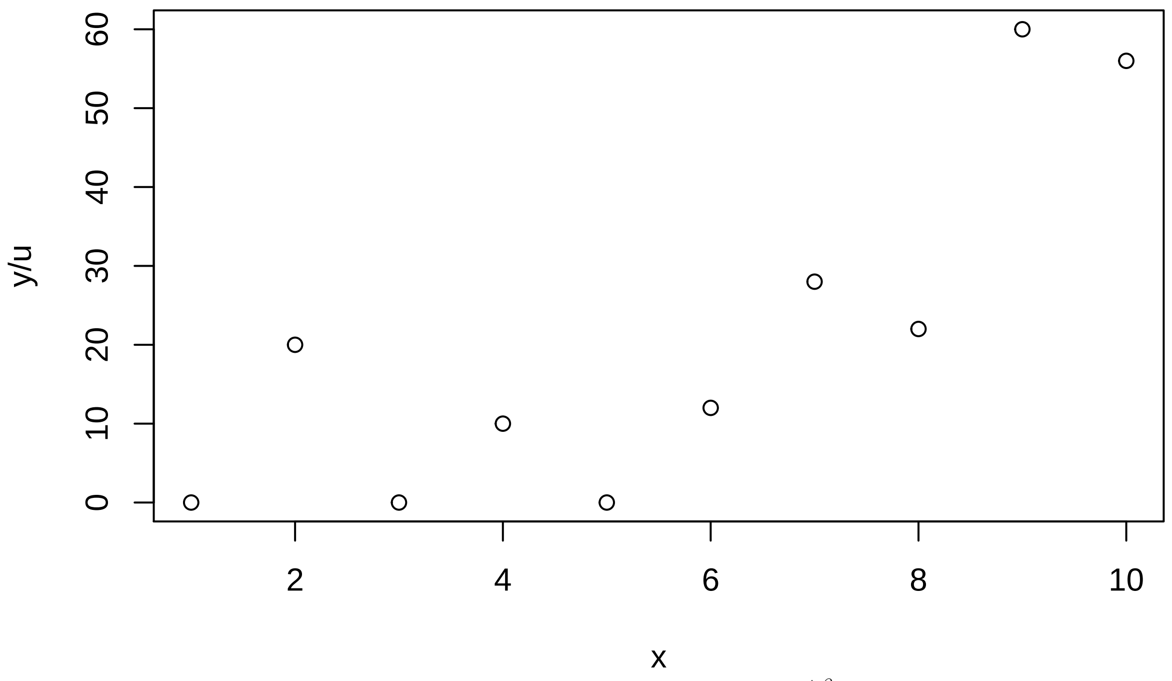
HW4

1. (a)

```r
n <- 10
x <- 1:10
u <- rep(c(0.1,0.5),c(5,5))
alpha <- 1
beta <- 0.3
y <- rpois(10, u * exp(alpha + (beta * x)))
plot(x, y/u)
```



(b)

The posterior density is given by $P(\alpha, \beta|\mathbf{y}, u_i) \propto \prod_i \text{Poisson}(y_i; u_i e^{\alpha+\beta x_i}\text{Unif})(\alpha; -\infty, \infty)\text{Unif}(\beta; -\infty, \infty)$ I estimated alpha and beta using a grid approximation over alpha from 0-5 and beta from 0-1 and checking 100 points in between:

```r
alphas <- numeric(10000)
betas <- numeric(10000)
posts <- numeric(10000)
i <- 1
norm <- 0
for(alpha in seq(0, 5, .05))
{
  for(beta in seq(0, 1, .01))
  {
    alphas[i] <- alpha
    betas[i] <- beta
    posts[i] <- exp(sum(log(dpois(y, u * exp(alpha + (beta * x))))))
    norm <- norm + posts[i]
    i <- i + 1
  }
}
posts <- posts/norm
```

I then computed the expectations and the variances: Alpha has a mean of:

```
print(sum(alphas * posts))
```

```
## [1] 0.7931087
```

and a variance of:

```
print(sum(alphas^2 * posts) - (sum(alphas * posts))^2)
```

```
## [1] 0.2087864
```

Beta has a mean of:

```
print(sum(betas * posts))
```

```
## [1] 0.3327095
```

and a variance of:

```
print(sum(betas^2 * posts) - (sum(betas * posts))^2)
```

```
## [1] 0.002880286
```

c. To find the posterior mode, I found the highest value in the approximated density. Using this method, I found that the posterior mode was at alpha =

```
df <- data.frame(alphas, betas, posts)
df[order(-posts),][1,]$alphas
```

```
## [1] 0.65
```

and beta =

```
df[order(-posts),][1,]$betas
```

```
## [1] 0.35
```

Then I numerically evaluated the hessian of the function to compute the observed information and inverted the matrix to get a covariance matrix of:

```
library(numDeriv)
```

```
## Warning: package 'numDeriv' was built under R version 3.1.3
```

```
library(pryr)
```

```
## Warning: package 'pryr' was built under R version 3.1.3
```

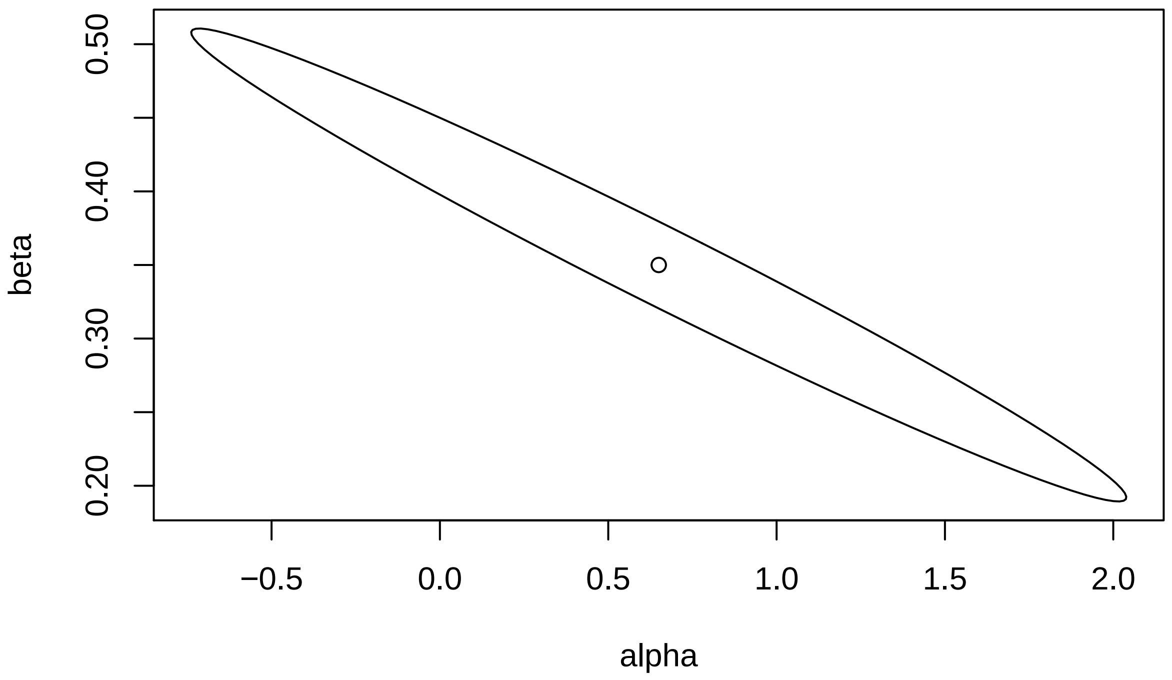
```
post <- function(x, y){
  x_data <- 1:10
  u <- rep(c(0.1,0.5),c(5,5))
  sum(log(dpois(y, u * exp(x[1] + (x[2] * x_data)))))
}
alpha = df[order(-posts),][1,]$alphas
beta = df[order(-posts),][1,]$betas
part_post <- partial(post, y=y)
I_theta <- - hessian(part_post, x=c(alpha, beta))
cov <- solve(I_theta)
print(cov)
```

```
##             [,1]         [,2]
## [1,]  0.3219584 -0.036607298
## [2,] -0.0366073  0.004308046
```

```
library(ellipse)
```

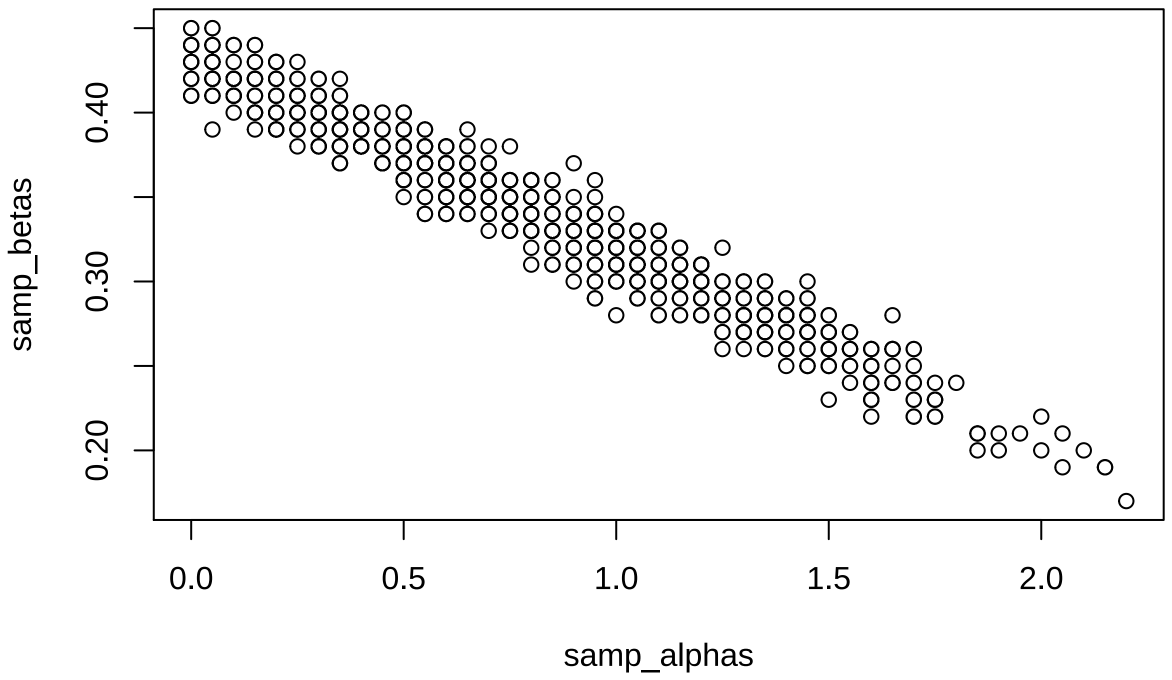
```
## Warning: package 'ellipse' was built under R version 3.1.2
```

```
plot(ellipse(cov, centre = c(alpha, beta)), type = 'l', xlab="alpha", ylab="beta",)
points(alpha, beta)
```



d.

```r
samp_alphas <- numeric(1000)
samp_betas <- numeric(1000)
post <- data.frame(alphas, betas, posts)
for(draw in 1:1000)
{
  prob = runif(1)
  p = 0
  i = 1
  while(p <= prob)
  {
    p = p + post[i,]$posts
    i = i + 1
  }
  samp_alphas[draw] <- post[i,]$alphas
  samp_betas[draw] <- post[i,]$betas
}
plot(samp_alphas, samp_betas)
```



2. a. We know that for a general continuous distribution the quantile estimate is asymptotically normal and the variance of the quantile statistic is given by $\frac{p(1-p)}{n(f(F^{-1}(p)))^2}$. Plugging in, we get $\frac{(0.975)(.025)}{n(f(\text{qnorm}(0.975)))^2} = \frac{0.024}{n(0.0034)}$. If we assume that the SD is 1 then we set the square root of that variance equal to .1 and get an n of 705

b. I simulated drawing 705 samples from a normal 100000 times and computed the SD. The empirical SD was:

```r
percentiles <- numeric(10000)
for(i in 1:length(percentiles))
{
  samps <- rnorm(705)
  percentiles[i] <- quantile(samps, probs = .975)
}
print(sd(percentiles))
```

```
## [1] 0.09869728
```

3. I used the Stan model and code from the paper but multiplied the number of x's by 25, 50 and 75, i.e.

```r
x <- rep((1:1000)/100, 25)
x <- rep((1:1000)/100, 50)
x <- rep((1:1000)/100, 75)
```

This converged with 75 but not with 25 or 50 suggesting that it converges between 50 and 75 times the number of x data.