hw4

Michael Pena

2024-12-10

problem 13

```r
# setting up data
bike <- read.csv("bike-data.csv", header = T)
```

```r
# in class code with MH and gibbs sampling
# predef the bayes functions
Prior <- function(a,b){
   (a + b)^(-5/2)
}

LLH <- function(theta,a,b){
    sum(log(dbeta(theta,a,b)))
}

Proposal <- function(a,b){ #Jacobian
    1/(a*b)
}

rProposal <- function(n,mean,cov){
    rmvnorm(n,mean,cov)
}

# build a function just for this algorithm
MHGIBBs <- function(y,N,B,alpha0,beta0,S.tune = diag(2)){
    # initializations
    J = length(y)
    accept = 0
    alpha.post = beta.post = numeric()
    theta.post = matrix(0,J,B)
    theta0 <- numeric(length = J)

    #loop
    for(b in 1:B){
        # Gibbs Step for theta
        for(j in 1:J){
        shp1 =  alpha0 + y[j]
        shp2 = beta0 + N[j] - y[j]
        theta0[j] = rbeta(1, shp1, shp2)
        }
        # Metro-Haste step for alpha and beta
        phi1 = rProposal(1, c(log(alpha0),log(beta0)), 1*S.tune)

        alpha1 =  exp(phi1[1])
        beta1 = exp(phi1[2])

        r = exp(
        LLH(theta0,alpha1,beta1)
        + log(Prior(alpha1,beta1))
        + log(Proposal(alpha0,beta0))
        - LLH(theta0,alpha0,beta0)
        - log(Prior(alpha0,beta0))
        - log(Proposal(alpha1,beta1)))
```

```r
        ## accept check
        if(runif(1) < min(1,r)){
        alpha0 = alpha1
        beta0 = beta1
        accept = accept + 1
        }
        # drop off the samplings
        alpha.post[b] <- alpha0
        beta.post[b] <- beta0
        theta.post[,b] <- theta0
        }
    # tuning the covariance matrix
    S.tune <- matrix(0,2,2)
        S.tune[2,1] <- S.tune[1,2] <- cov(log(alpha.post),log(beta.post))
    S.tune[1,1] <- var(log(alpha.post))
    S.tune[2,2] <- var(log(beta.post))

     print(accept/B)
    # attributes in the function
    return(list("alpha" = alpha.post, "beta" = beta.post, "theta" = theta.post, "AR" = accept/B, "S" = S

}
```

```r
# let's run our functions
y <- bike$Bicycles
N <- bike$Bicycles + bike$OtherVehicles
mean(y/N) # this is about .2, make alpha0 = 2, beta0 = 8
```

(b)

```
## [1] 0.1961412
```

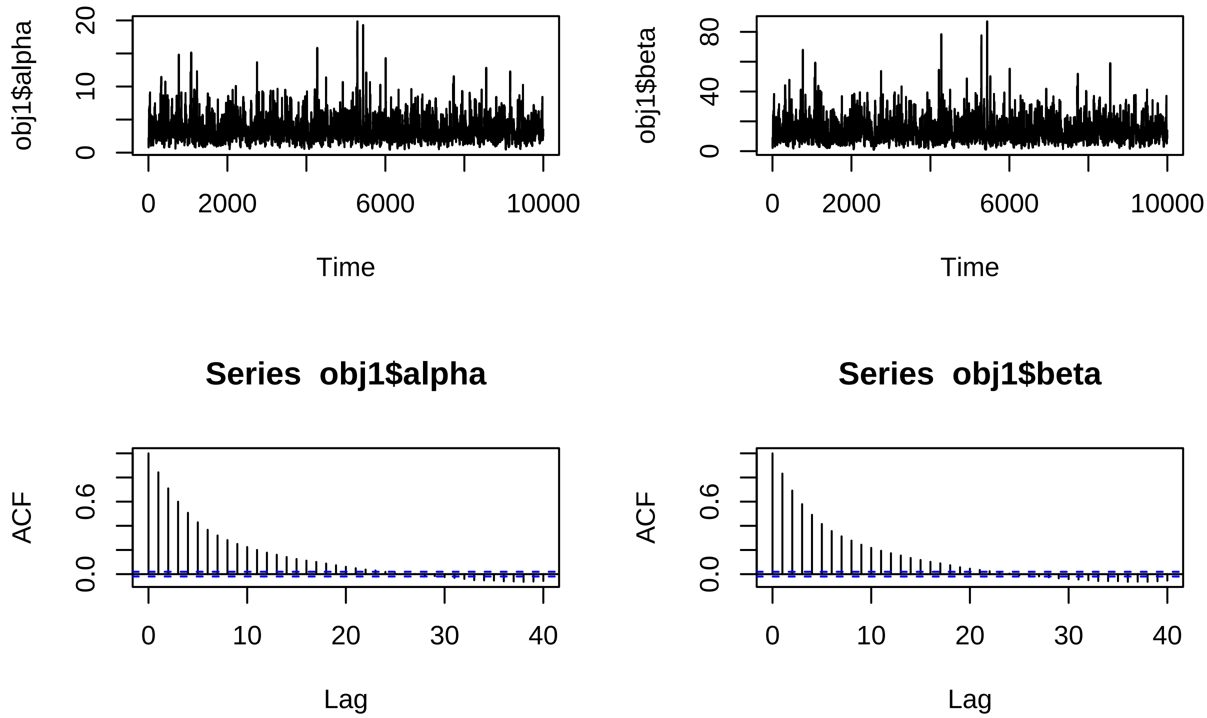
```r
MHGIBBs(y,N,5000,2,8)$S -> S1 # run 1 time to get tuning matrix
```

```
## [1] 0.1184
```

```r
MHGIBBs(y,N,10000,2,8,S1) -> obj1
```

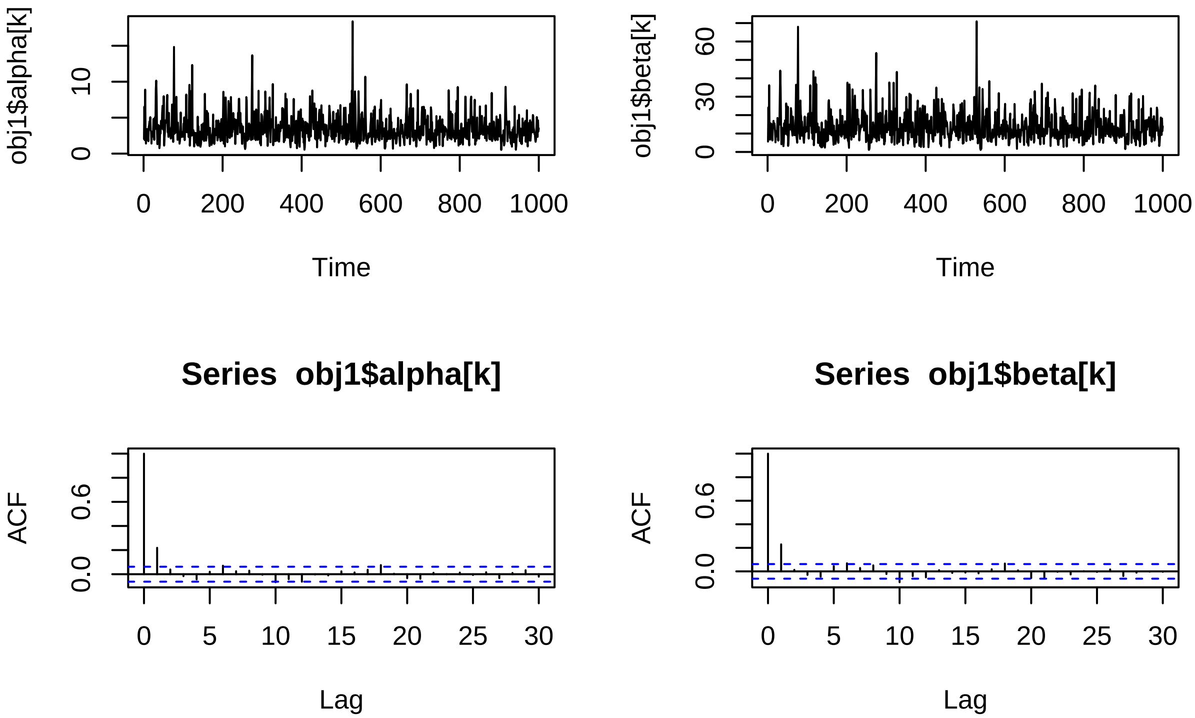
```
## [1] 0.5126
```

```r
# visualizations
par(mfrow = c(2,2))
plot.ts(obj1$alpha)
plot.ts(obj1$beta)
acf(obj1$alpha)
acf(obj1$beta)
```

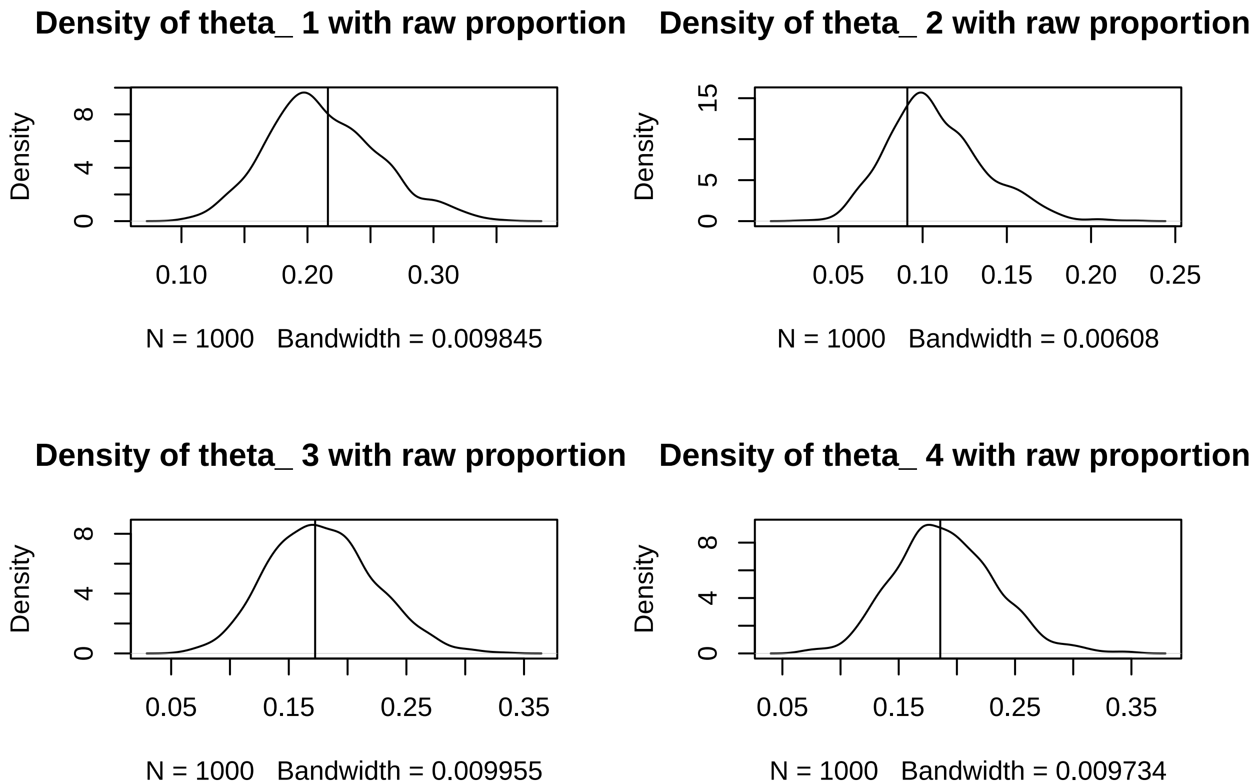



```
# take out all but the 10th lag
k = 10*(1:10000)
k = k[k <= 10000]
```

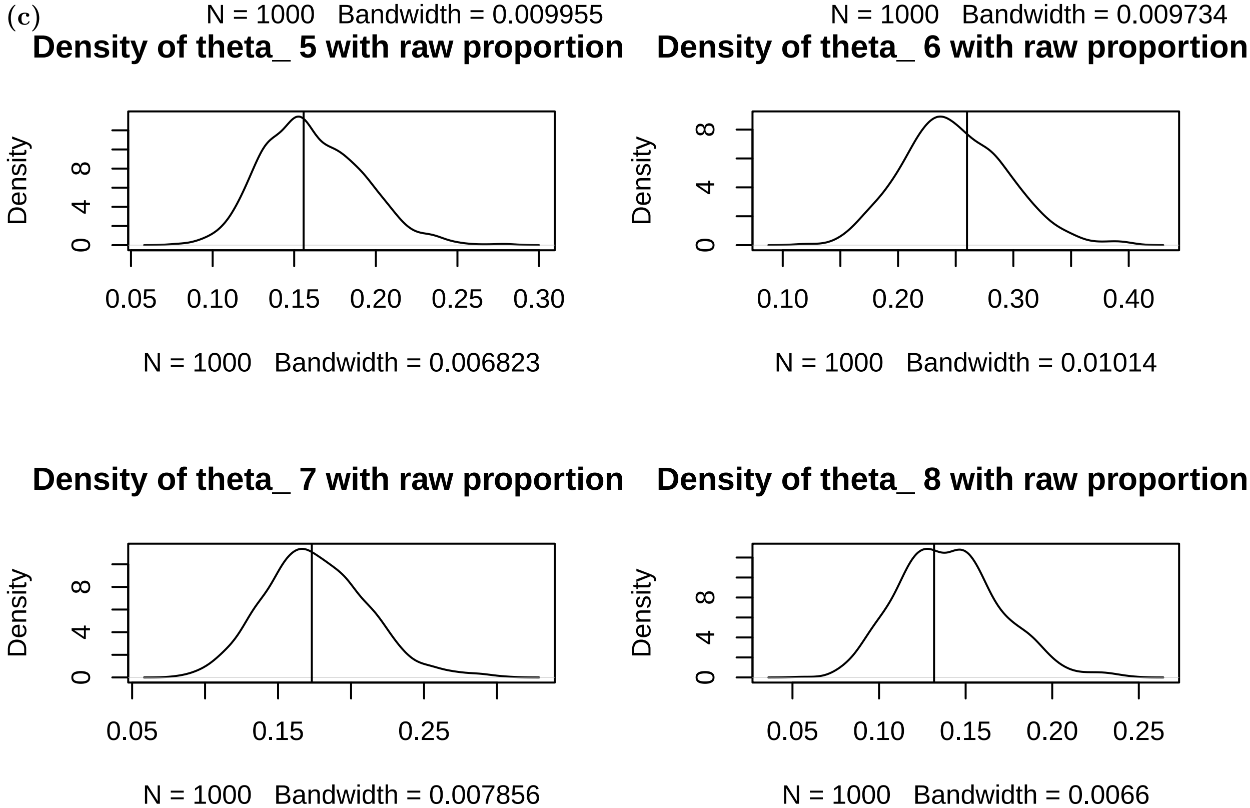
```
# visualizations
par(mfrow = c(2,2))
plot.ts(obj1$alpha[k])
plot.ts(obj1$beta[k])
acf(obj1$alpha[k])
acf(obj1$beta[k])
```

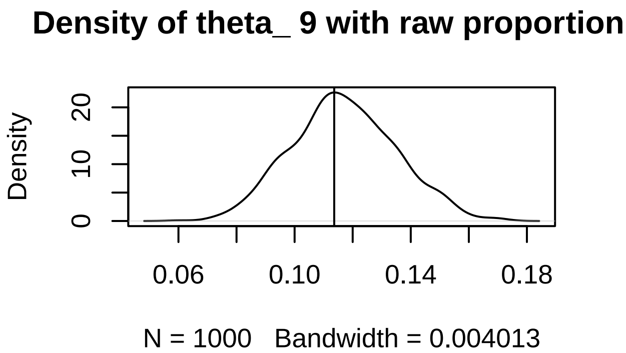


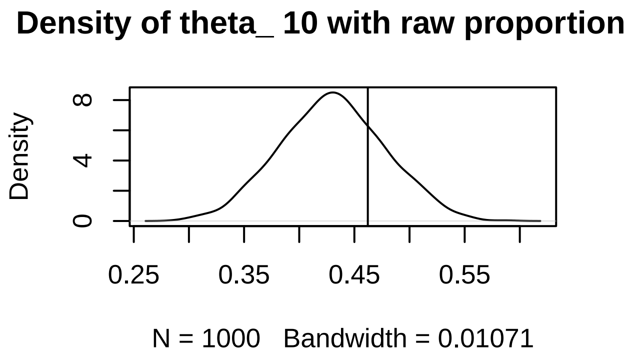

```
par(mfrow = c(2,2))
for(d in 1:10){
  plot(density(obj1$theta[d,k]),main = paste("Density of theta_",d, "with raw proportion",d))
  abline(v = (y/N)[d])
}
```


Density of theta_ 1 with raw proportion
Density
N = 1000 Bandwidth = 0.009845
Density of theta_ 2 with raw proportion
Density
N = 1000 Bandwidth = 0.00608
Density of theta_ 3 with raw proportion
Density
N = 1000 Bandwidth = 0.009955
Density of theta_ 4 with raw proportion
Density
N = 1000 Bandwidth = 0.009734

(c)

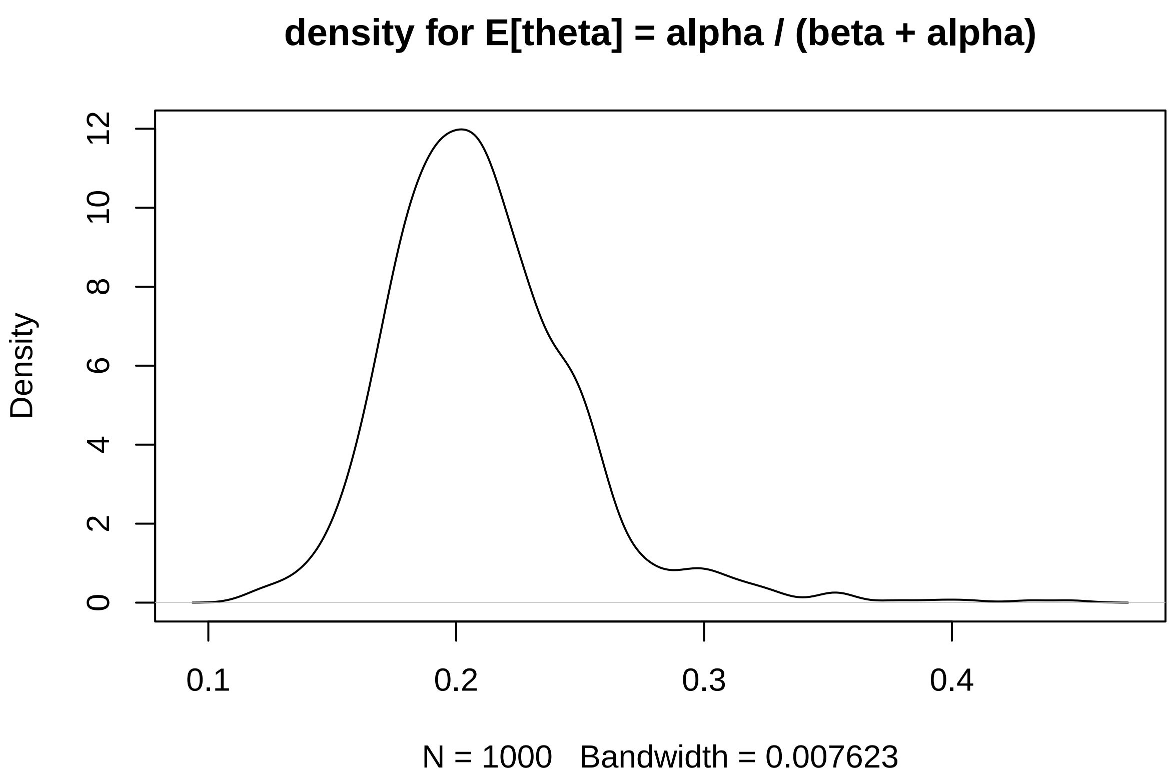

Density of theta_ 5 with raw proportion
Density
N = 1000 Bandwidth = 0.006823
Density of theta_ 6 with raw proportion
Density
N = 1000 Bandwidth = 0.01014
Density of theta_ 7 with raw proportion
Density
N = 1000 Bandwidth = 0.007856
Density of theta_ 8 with raw proportion
Density
N = 1000 Bandwidth = 0.0066





The MAPs of the simulated inferences from the posterior distribution are not too far off from the raw data proportions and so I think there is reality to the model that we've derived.

```
alp <- obj1$alpha[k]
bet <- obj1$beta[k]
expec <- alp/(bet + alp)
plot(density(expec), main = "density for E[theta] = alpha / (beta + alpha)")
```



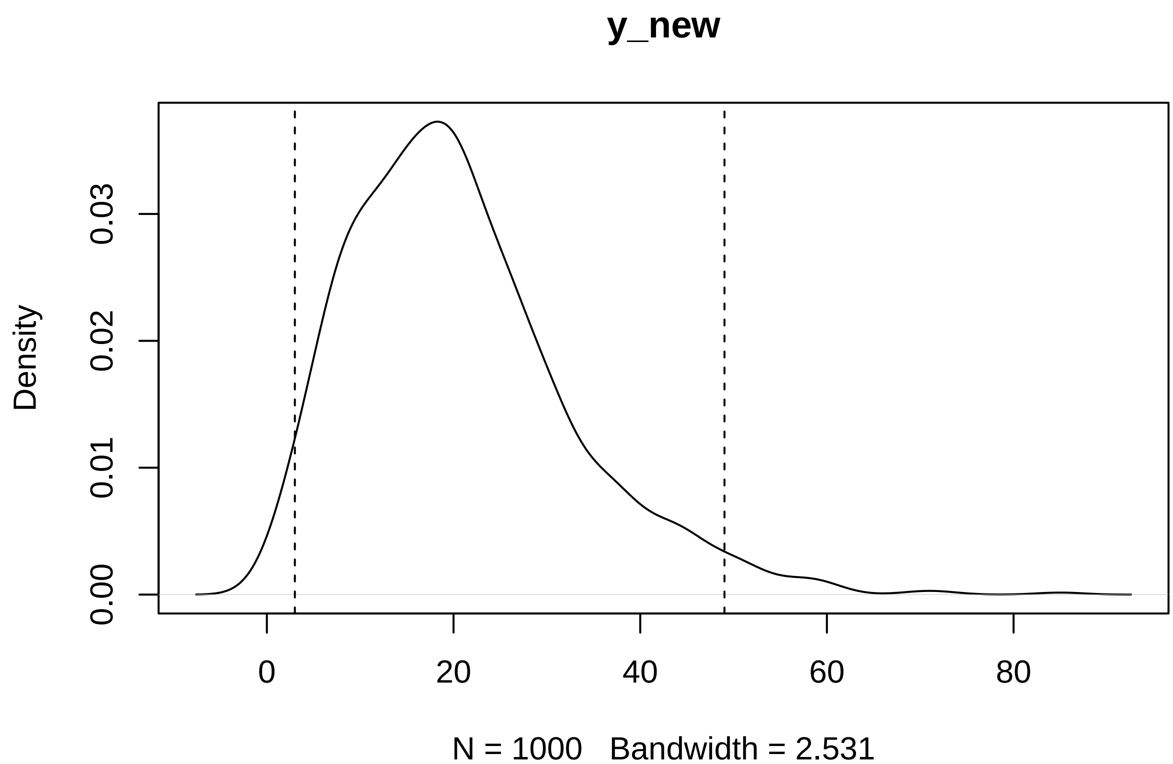
(d)

```
quantile(expec, c(0.025,.975))
```

```
##      2.5%     97.5%
## 0.1507044 0.3016990
```

```
# input alphas and betas into a beta distribution
rbeta(1000,obj1$alpha[k],obj1$beta[k]) -> newtheta
rbinom(1000,100,newtheta) -> newy
```

```
plot(density(newy), main = "y_new")
quantile(newy,c(.025,.975)) -> q1
abline(v=q1, lty = 2)
```



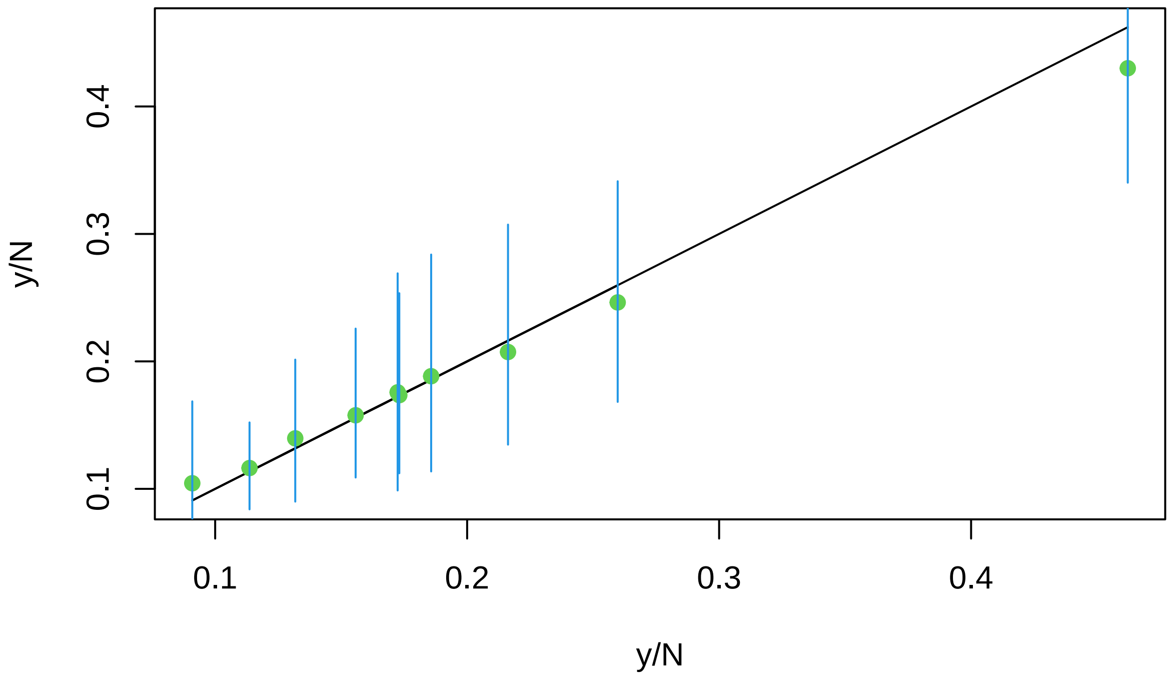
(e)

```
q1
```

```
##   2.5%  97.5%
##  3.000 49.025
```

I am not sure I can trust this confidence interval and the density skews right and this might not look good to city planners who I am consulting.

```
# checking analytically if this makes sense
CI = matrix(0, ncol = 3, nrow = 10)
for(j in 1:10){
  CI[j,] = quantile(obj1$theta[j,k], probs = c(0.025,0.5, 0.975))
}
plot(y/N, y/N, type = "l")
points(y/N, CI[,2], pch = 19, col = 3)
for(j in 1:10){
  points(c(y[j]/N[j],y[j]/N[j]),  c(CI[j,1],CI[j,3]), type ="l", col = 4)
}
```



(f)

This distribution is pretty reasonable according to this graph as it is diagonal.

Additional Problem

```r
# setup the data
schools <- read.csv(file = "schools.csv", header = T)
schools <- list("J" = 8,
                "y" = schools$estimate,
                "sigma" = schools$sd)
```

```r
# run the STAN and fit the data
# schools_fit <- stan(file="schools.stan",
#                      data = schools,
#                      iter = 1000,
#                      chains = 4)
fit1 <- stan(
  file = "schools.stan", # Stan program
  data = schools,    # named list of data
  chains = 4,             # number of Markov chains
  warmup = 1000,          # number of warmup iterations per chain
  iter = 20000,           # total number of iterations per chain
  cores = 2,              # number of cores
  refresh = 1000,         # show progress every 'refresh' iterations
  thin = 10               # number of thinning
)
```

```
## Trying to compile a simple C file
```

```
## Running /usr/lib/R/bin/R CMD SHLIB foo.c
## using C compiler: ‘gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0’
## gcc -I"/usr/share/R/include" -DNDEBUG   -I"/home/cern/R/x86_64-pc-linux-gnu-library/4.4/Rcpp/include/
## In file included from /home/cern/R/x86_64-pc-linux-gnu-library/4.4/RcppEigen/include/Eigen/Core:19,
##                  from /home/cern/R/x86_64-pc-linux-gnu-library/4.4/RcppEigen/include/Eigen/Dense:1,
```

```
##                   from /home/cern/R/x86_64-pc-linux-gnu-library/4.4/StanHeaders/include/stan/math/pri
##                   from <command-line>:
## /home/cern/R/x86_64-pc-linux-gnu-library/4.4/RcppEigen/include/Eigen/src/Core/util/Macros.h:679:10: 
##   679 | #include <cmath>
##       |          ^~~~~~~
## compilation terminated.
## make: *** [/usr/lib/R/etc/Makeconf:195: foo.o] Error 1
```

```
## Warning: There were 5 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```r
print (fit1)
```

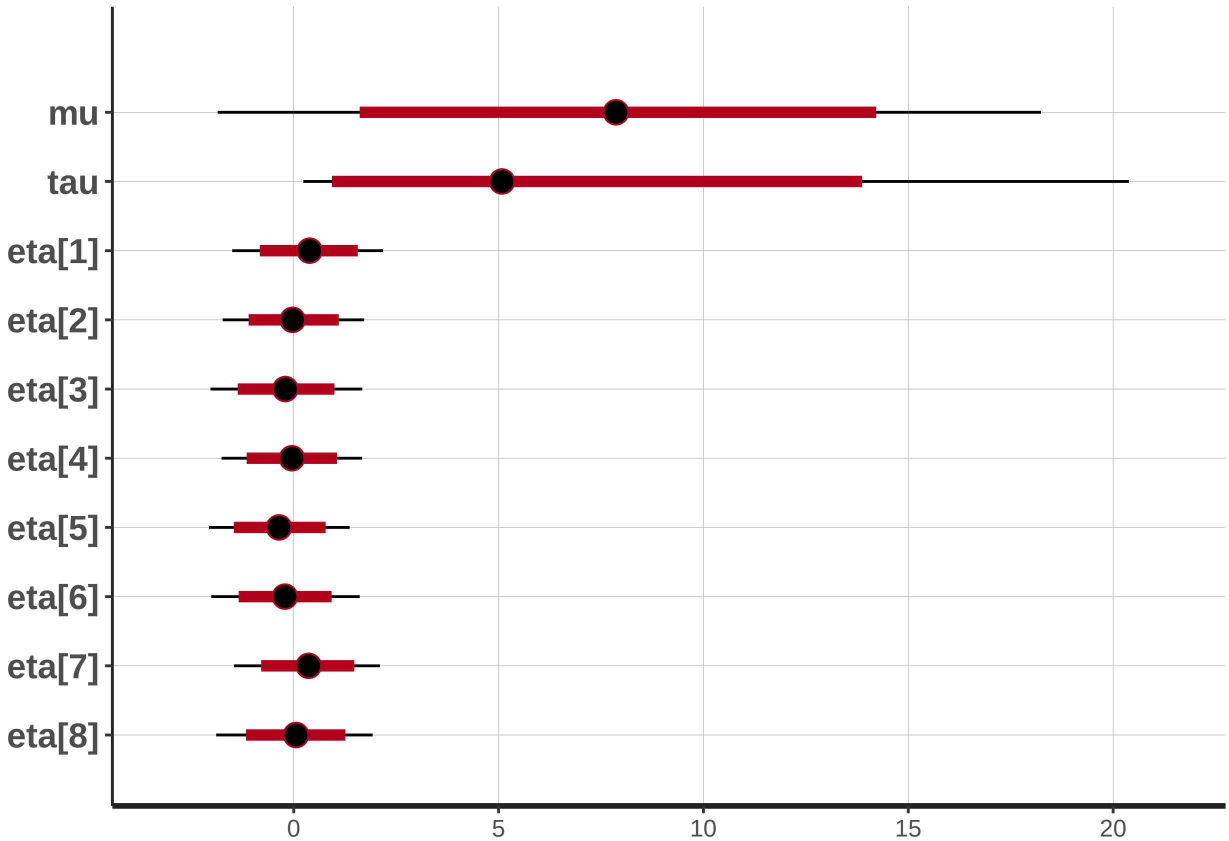
```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=20000; warmup=1000; thin=10;
## post-warmup draws per chain=1900, total post-warmup draws=7600.
##
##           mean se_mean   sd   2.5%   25%   50%   75% 97.5% n_eff Rhat
## mu        7.93    0.06 5.14  -1.85  4.58  7.86 11.22 18.24  7125    1
## tau       6.51    0.06 5.56   0.24  2.41  5.09  9.07 20.39  7402    1
## eta[1]    0.39    0.01 0.94  -1.50 -0.23  0.39  1.02  2.18  7038    1
## eta[2]   -0.01    0.01 0.87  -1.74 -0.58 -0.03  0.54  1.72  8032    1
## eta[3]   -0.19    0.01 0.93  -2.03 -0.80 -0.20  0.41  1.67  7778    1
## eta[4]   -0.04    0.01 0.87  -1.77 -0.63 -0.04  0.54  1.67  7466    1
## eta[5]   -0.36    0.01 0.88  -2.07 -0.94 -0.36  0.22  1.37  7345    1
## eta[6]   -0.21    0.01 0.90  -2.01 -0.80 -0.22  0.36  1.61  7662    1
## eta[7]    0.35    0.01 0.89  -1.46 -0.23  0.36  0.94  2.11  7551    1
## eta[8]    0.06    0.01 0.95  -1.89 -0.56  0.06  0.69  1.93  7397    1
## theta[1] 11.30    0.10 8.36  -2.25  5.86 10.16 15.43 31.61  7433    1
## theta[2]  7.79    0.07 6.25  -4.71  3.88  7.78 11.66 20.24  7567    1
## theta[3]  6.10    0.09 7.95 -11.85  1.96  6.65 10.94 20.69  7512    1
## theta[4]  7.58    0.08 6.54  -5.99  3.63  7.59 11.64 20.66  7443    1
## theta[5]  5.08    0.07 6.28  -8.79  1.34  5.53  9.38 16.12  7865    1
## theta[6]  6.19    0.08 6.64  -8.13  2.35  6.42 10.44 18.76  7493    1
## theta[7] 10.70    0.08 6.86  -1.00  6.05 10.00 14.67 26.18  7640    1
## theta[8]  8.44    0.09 7.96  -7.20  3.73  8.18 12.80 25.46  7194    1
## lp__     -4.93    0.03 2.69 -10.86 -6.53 -4.72 -3.05 -0.38  7856    1
##
## Samples were drawn using NUTS(diag_e) at Tue Dec 10 13:45:31 2024.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```r
plot (fit1)
```

```
## 'pars' not specified. Showing first 10 parameters by default.
```

```
## ci_level: 0.8 (80% intervals)
```

```
## outer_level: 0.95 (95% intervals)
```




```r
# traceplot
traceplot(fit1, pars = c("mu", "tau"), inc_warmup = T, nrow = 2)
```