Homework3

Problem 1

Let

$$Y_1, Y_2, ..., Y_n \sim N(\mu, \sigma^2)$$

For now assume $\sigma = 15, \bar{y} = 113, n = 10$

We have

$$p(\mu) = N(\mu_0, \sigma_0^2) = N(100, 15)$$

We know from the slides that

$$E(\mu|y) = \frac{\frac{\mu_0}{15} + \frac{n\bar{y}}{15}}{\frac{n+1}{15}}$$

We can see that the σ terms drop out and we are left with

$$E(\mu|y) = \frac{\mu_0 + n\bar{y}}{n+1}$$

Since we know that $\mu|y$ is normally distributed we can construct a confidence interval based on

$$\frac{\mu_0 + n\bar{y}}{n+1} - 1.96 * \sigma, \frac{\mu_0 + n\bar{y}}{n+1} + 1.96 * \sigma$$

where

$$\sigma = \frac{1}{\frac{1}{15} + \frac{n}{15}} = \frac{15}{n+1} = V$$

Problem 2

We know by definition that bias is

$$E(\hat{\mu}|\mu^*) - \mu^*$$

Let's take a closer look at

$$E(\hat{\mu}|\mu^*)$$

$$= E(\frac{\mu_0 + n\bar{y}}{n+1}|\mu*) = \frac{\mu_0}{n+1} + \frac{n}{n+1}\mu*$$

We can see that the Bayesian estimator has higher bias than the frequentist estimator.

We also need to examine

$$Var(\hat{\mu}|\mu^*) = Var(\frac{\mu_0 + n\bar{y}}{n+1}|\mu*) = Var(\frac{n\bar{y}}{n+1}|\mu*)$$

because the first patrt is a constant.

$$= (\frac{n}{n+1})^2\sigma^2$$

We can see that the Bayesian estimator has a smaller variance than the frequentist estimator by $n/n+1 < 1$ We can now consider the MSE of the bayesian estimator

$$= (\frac{n}{n+1})^2\sigma^2 + (\frac{\mu_0}{n+1} + \frac{n}{n+1}\mu*)^2$$

We know by the properties of the sampling distribution that the frequentist MSE is

$$\frac{\sigma^2}{n} + 0^2$$

Lets look at

$$(\frac{n}{n+1})^2\sigma^2 + (\frac{\mu_0}{n+1} + \frac{n}{n+1}\mu*)^2 - \frac{\sigma^2}{n}?0$$

We know that bias^2 term is >0 so we just need to ask

$$(\frac{n}{n+1})^2\sigma^2 - \frac{\sigma^2}{n}?0$$

$$\sigma^2((\frac{n}{n+1})^2 - \frac{1}{n})?0$$

$$\sigma^2((\frac{n^3}{n(n+1)^2}) - \frac{(n+1)^2}{n(n+1)^2})?0$$

$$\sigma^2(\frac{n^3 - (n^2+2n+1)}{n(n+1)^2})?0$$

$$\sigma^2(\frac{n^3 - n^2 - 2n - 1)}{n(n+1)^2})?0$$

which for $n > 2$ is strictly positive.

```
library(foreign)
# install JAGS from http://sourceforge.net/projects/mcmc-jags/files/
# and load these packages:
library(rjags)
```

```
## Loading required package: coda
```

```
## Linked to JAGS 4.2.0
```

```
## Loaded modules: basemod,bugs
```

```
library(R2jags)
```

```
##
## Attaching package: 'R2jags'
```

```
## The following object is masked from 'package:coda':
##
##     traceplot
```

```
# if you want to use the stan shiny app for diagnostics:
kidiq <- read.dta("/home/gcgibson/bayes2018/kidiq (1).dta")
names(kidiq)
```

```
## [1] "kid_score" "mom_hs"    "mom_iq"    "mom_work"  "mom_age"
```

```
y.i <- rnorm(100,100,13)#kidiq$kid_score
n <- length(y.i)


model <-
  "model{
for (i in 1:n){
y.i[i] ~ dnorm(mu, tau.y)
# note: normals in JAGS use a precision parameter!!
}
tau.y ~ dgamma(nu0/2, nu0/2*sigma.y0^2)
mu ~ dnorm(mu0, tau.mu0) # again, use precision!
}"

# prior parameters
mu0 <- 100
sigma.mu0 <- 15
nu0 <- 1
sigma.y0 <- 15

jags.data <- list(
  # elements below have been defined already
  y.i = y.i,
  n = n,
  tau.mu0 = 1/sigma.mu0^2,
  mu0 = mu0,
  nu0 = nu0,
  sigma.y0   = sigma.y0
  #  tau.0  = 1/sigma.y0^2
)

parnames <- c("mu", "tau.y")
mod<-jags(data = jags.data,
          parameters.to.save=parnames,
          model.file = textConnection(model))
```

```
## module glm loaded
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 100
##    Unobserved stochastic nodes: 2
##    Total graph size: 113
##
## Initializing model
```

```r
print(mod)
```

```
## Inference for Bugs model at "5", fit using jags,
##  3 chains, each with 2000 iterations (first 1000 discarded)
##  n.sims = 3000 iterations saved
##          mu.vect sd.vect    2.5%     25%     50%     75%   97.5%  Rhat
## mu       100.150   1.344  97.589  99.246 100.144 101.066 102.833 1.001
## tau.y      0.006   0.001   0.004   0.005   0.006   0.006   0.008 1.003
## deviance 799.248   2.034 797.244 797.786 798.647 800.036 804.651 1.001
##          n.eff
## mu        3000
## tau.y      850
## deviance  2800
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 2.1 and DIC = 801.3
## DIC is an estimate of expected predictive error (lower deviance is better).
```

```r
names(mod)
```

```
## [1] "model"              "BUGSoutput"         "parameters.to.save"
## [4] "model.file"         "n.iter"             "DIC"
```

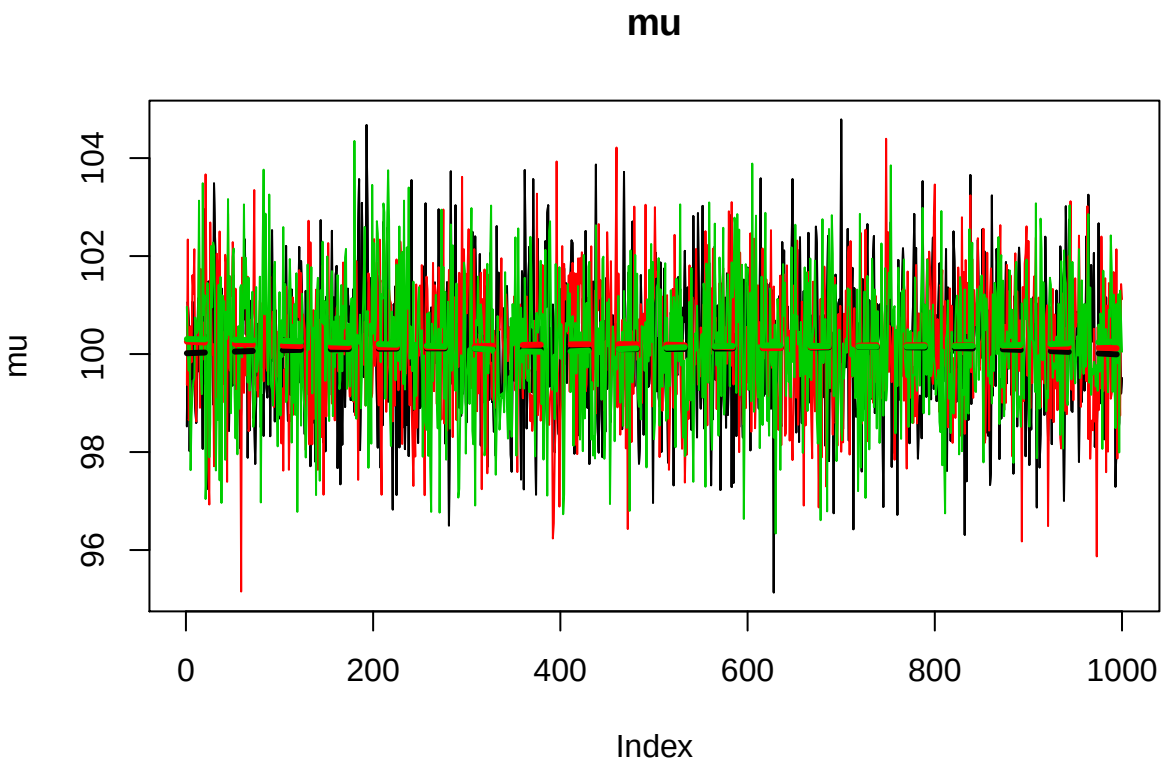
```r
# samples are saved in sims.array, which has dimension niterationsxnchainsx(nparameters+deviance)
mcmc.array <- mod$BUGSoutput$sims.array
dim(mcmc.array)
```

```
## [1] 1000    3    3
```

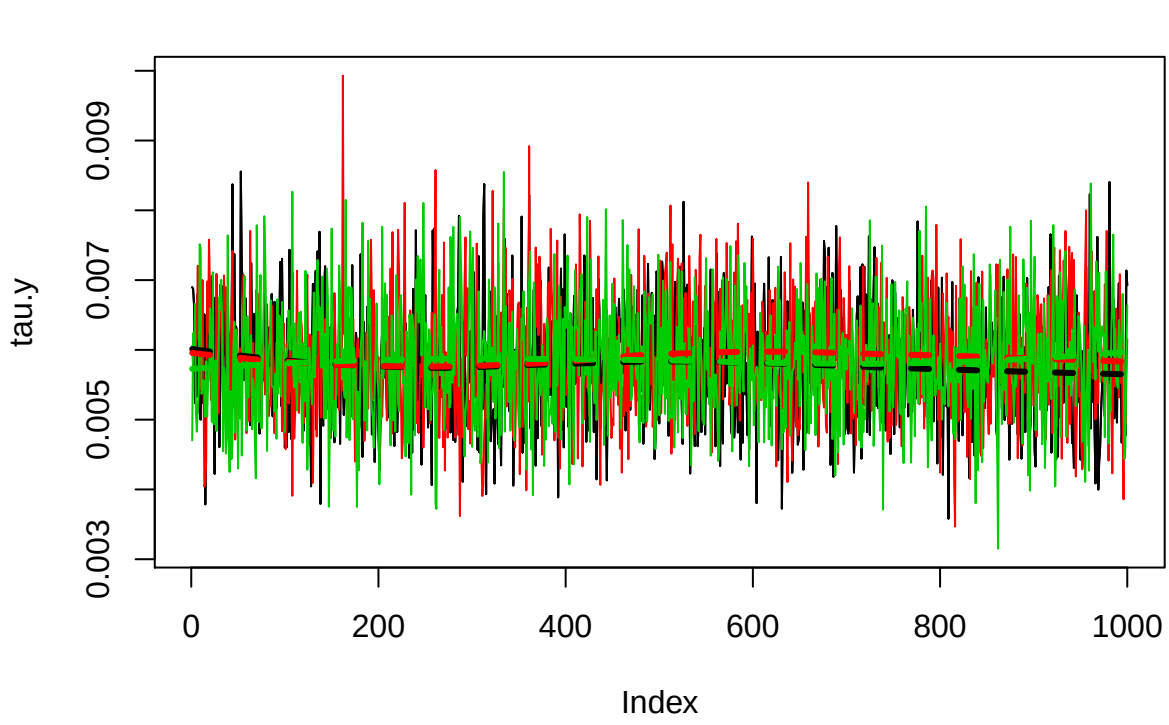
```r
# default is 2000 iterations of which 1000 are burnin.
# 3 chains and here we have 2 parameters

# I usually use a little function to make traceplots
#-------------------------------------------------
PlotTrace <- function(#Traceplot for one parameter
  ### Trace plot for one parameter and add loess smoother for each chain
  parname, mcmc.array,##<< needs to be 3-dimensional array!
  n.chains= NULL, n.sim= NULL, main = NULL){
  if (is.null(main)) main <- parname
  if (is.null(n.sim)) n.sim <- dim(mcmc.array)[1]
  if (is.null(n.chains)) n.chains <- dim(mcmc.array)[2]
  plot(c(mcmc.array[,1,parname]), type = "l", ylab = parname,  main = main,
       ylim = c(min(mcmc.array[,,parname]),max(mcmc.array[,,parname])))
  for (chain in 1:n.chains){
    lines(c(mcmc.array[,chain,parname]), type = "l", col = chain)
  }
  for (chain in 1:n.chains){
    curve(predict(loess(c(mcmc.array[,chain,parname])~seq(1,n.sim)),x), lty = 2, lwd = 3, add = TRUE, ty
  }
}
#-------------------------------------------------
#pdf(paste(figdir, "jags1.pdf", sep =""), width = 6, height = 4)
#par(mfrow = c(1,2), lwd = 3, cex.axis = 2, cex.lab = 2, cex.main = 2,mar = c(5,5,3,3))
```

```
PlotTrace(parname = "mu", mcmc.array)
```




```
PlotTrace(parname = "tau.y", mcmc.array)
```




```
#dev.off()
```

```r
mu.s <-  c(mcmc.array[,,"mu"])
mean(mu.s)
```

```
## [1] 100.1498
```

```r
quantile(mu.s, c(0.025, 0.975))
```

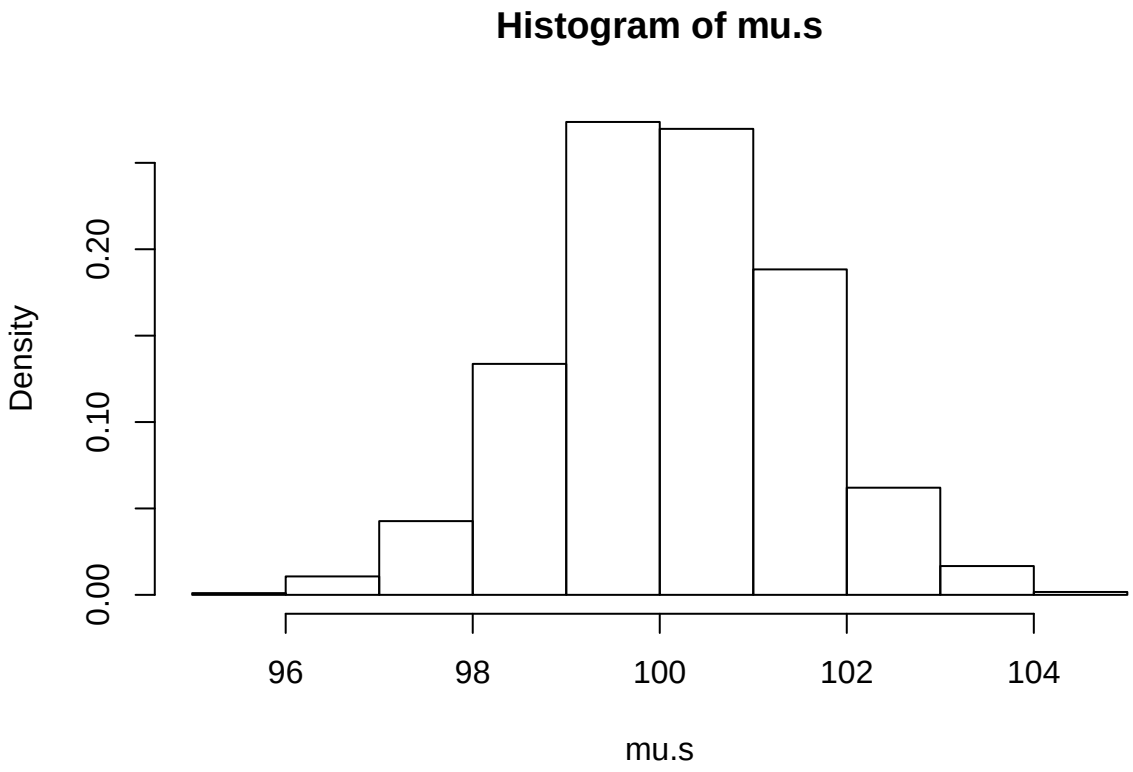
```
##      2.5%     97.5%
##  97.58868 102.83274
```

```r
# or directly from mod
round(mod$BUGSoutput$summary["mu", c("mean", "2.5%", "97.5%", "n.eff", "Rhat")],2)
```

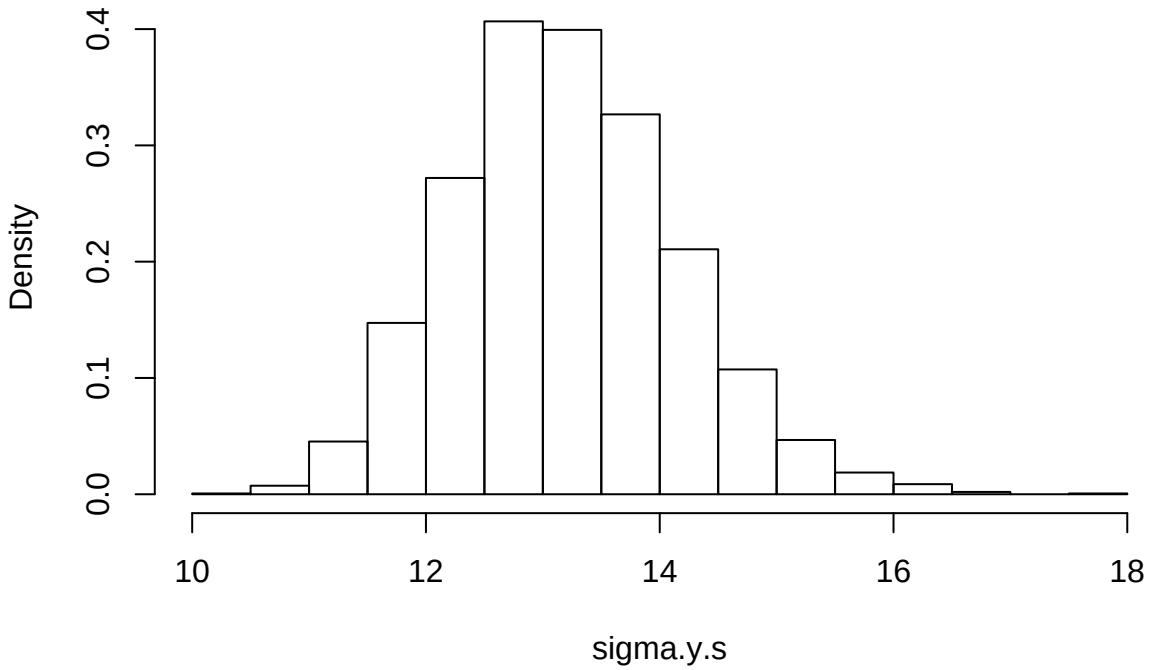
```
##    mean    2.5%   97.5%   n.eff    Rhat
##  100.15   97.59  102.83 3000.00    1.00
```

```r
sigma.y.s <- 1/sqrt(c(mcmc.array[,,"tau.y"]))

hist(mu.s, freq = F)
```




```r
hist(sigma.y.s, freq = F)
```




```
print (mean(sigma.y.s))
```

```
## [1] 13.20112
```

```
print (quantile(sigma.y.s, c(0.025, 0.975)))
```

```
##     2.5%    97.5%
## 11.48616 15.19913
```