Lab 8, STA 360/602

Rebecca C. Steorts

The model is now

$$
\begin{aligned}
Y_i \mid Z_i, \mu_1, \mu_2, \mu_3, \epsilon^2 &\sim N(\mu_{Z_i}, \epsilon^2) \\
\mu_j \mid \mu_0, \sigma_0^2 &\sim N(\mu_0, \sigma_0^2) \\
\mu_0 &\sim N(0, 3) \\
\sigma_0^2 &\sim IG(2, 2) \\
(w_1, w_2, w_3) &\sim Dirichlet(3, \mathbf{1}) \\
\epsilon^2 &\sim IG(2, 2) \\
Z_i \mid w_1, w_2, w_3 &\sim Cat(3, \boldsymbol{w}).
\end{aligned}
$$

The full conditionals were given in the lab solutions, and are reproduced below for reference:

$$
\begin{aligned}
p(\mu_0 \mid \dots) &= N\left(\frac{\sigma_0^2 \sum_{i=1}^3 \mu_i}{1/3 + 3\sigma_0^{-2}}, (1/3 + 3\sigma_0^{-2})^{-1}\right), \\
p(\sigma_0^2 \mid \dots) &= IG\left(2 + 3/2, 2 + (1/2)\sum_{i=1}^3 (\mu_i - \mu_0)^2\right), \\
p(\epsilon^2 \mid \dots) &= IG\left(2 + n/2, 2 + (1/2)\sum_{i=1}^n (Y_i - \mu_{Z_i})^2\right), \\
p(\boldsymbol{w} \mid \dots) &= Dir(3, (1 + N_1, 1 + N_2, 1 + N_3)), \\
p(\mu_j \mid \dots) &= N\left(\left(\mu_0\sigma_0^{-2} + \epsilon^{-2}\sum_{i:Z_i=j} y_i\right)(\sigma_0^{-2} + N_j\epsilon^{-2})^{-1}, \left(\sigma_0^{-2} + N_j\epsilon^{-2}\right)^{-1}\right), \\
P(Z_i = j) &= \frac{wjN(y_i \mid \mu_j, \epsilon^2)}{\sum_{k=1}^3 w_k N(y_i \mid \mu_k, \epsilon^2)}.
\end{aligned}
$$

The code for part of Gibbs' sampler is given below. (You will need to finish the code for your homework this week).

One thing to note about the sampler is that we do *not* save the sampled values for Z_i. One reason is that these variables were added to the model for convenience, so they are not necessarily of interest. In addition, the distribution of these variables is sufficiently summarized by $\boldsymbol{w}$, which we will have a distribution for. There is also a practical reason to discard these samples. We must sampler 500 values at each iteration. If we run only 1000 iterations (and we may reasonably desire to run 10 times this many iterations), we must store 500,000 values. This is a large amount of memory and could drastically affect the speed of the sampler.

```r
library(xtable)
library(MCMCpack)
```

```
## Warning: package 'MCMCpack' was built under R version 3.4.3
```

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

```
## Loading required package: MASS
## ##
## ## Markov Chain Monte Carlo Package (MCMCpack)
## ## Copyright (C) 2003-2018 Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park
## ##
## ## Support provided by the U.S. National Science Foundation
## ## (Grants SES-0350646 and SES-0350613)
## ##
```

```
categSampler <- function(probs){
  # this samples from a categorical distribution
  # the probabilites are given by probs
  cdf <- cumsum(probs)
  x <- runif(1)
  samp <- which(x <= cdf)[1]
  return(samp)
}
```

```
# set prior parameters
mu.mu <- 0
mu.v <- 3
s.a <- 2
s.b <- 2
e.a <- 2
e.b <- 2
```

```
nnUpdate <- function(prior.mu, prior.s, like.s, data){
  # computes the posterior parameters for a normal-normal model
  # prior.mu is prior mean
  # prior.s is prior variance
  # like.s is variance of likelihood
  # data is data observations
  x <- sum(data)
  n <- length(data)
  var <- 1/(1/prior.s + n/like.s)
  mu <- ((prior.mu/prior.s) + (x/like.s))*var
  return(c(mu,sqrt(var)))
}
```

```
nIGUpdate <- function(a, b, mu, data){
  # computes parameters for IG (shape, rate)
  # a is prior shape
  # b is prior rate
  # mu is mean of likelihood
  # data is data
  n <- length(data)
  a.post <- a + n/2
  b.post <- b + 1/2*sum((data-mu)^2)
  return(c(a.post,b.post))
}
```

```
counter <- function(k, zs){
  # counts the number of z_i = j for j = 1,..,k
  counts <- vector(mode = "numeric", length = k)
  for (i in 1:k){
```

```r
    counts[i] <- sum(zs == i)
  }
  return(counts)
}
```

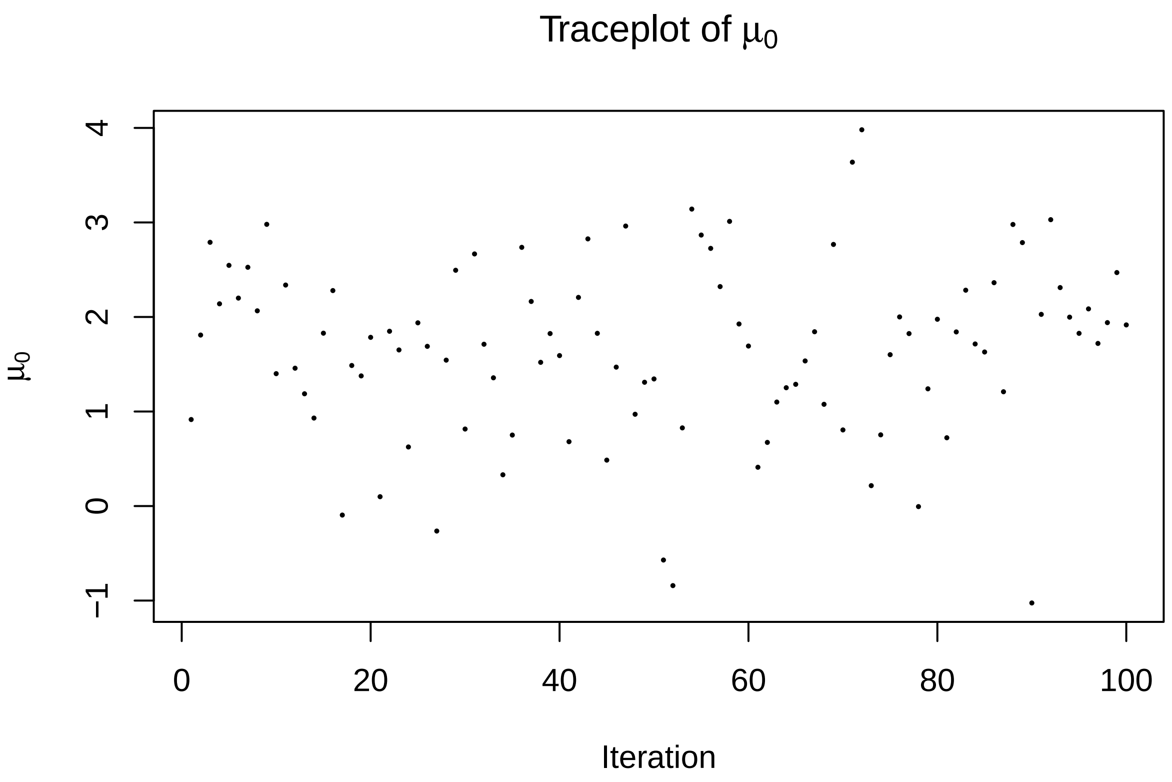
```r
catProbs <- function(w, mus, epsilon, y.i){
  # calculates the probabilites that define the categorical distribution
  # for Z in our sampler
  unnormed <- w*dnorm(y.i, mean = mus, sd = sqrt(epsilon))
  probs <- unnormed/sum(unnormed)
  return(probs)
}
```

```r
augSampler <- function(ys, zs, mus, m0, s0, w, epsilon, n.iter,
                       burnin = 1){
  # Gibbs sampler with augmented data model
  # ys is data
  # zs is initial values of z
  # mus is (mu_1,mu_2,mu_3) initial values
  # m0 is initial value for mu_0
  # s0 is initial value for sigma_0^2
  # w is vector (w_1,w_2,w_3) that sums to one, initial value
  # epsilon is epsilon^2 initial value
  # n.iter is number of iterations
  # burnin is number of sampler to drop for burnin
  n <- length(ys)
  m <- length(mus)+length(w)+3
  k <- length(mus)
  res <- matrix(NA, nrow = n.iter, ncol = m)
  for (i in 1:n.iter){
    # update parameters, then sample
    m0.params <- nnUpdate(mu.mu, mu.v, s0, mus)
    m0 <- rnorm(1, mean = m0.params[1], m0.params[2])
    s0.params <- nIGUpdate(s.a,s.b, m0, mus)
    s0 <- 1/rgamma(1, shape = s0.params[1], rate = s0.params[2])
    ep.params <- nIGUpdate(e.a,e.b, mus[zs], ys)
    epsilon <- 1/rgamma(1, shape = ep.params[1], rate = ep.params[2])
    # calculate number in each category
    Ns <- counter(k, zs)
    w <- rdirichlet(1, 1 + Ns)
    for (j in 1:k){
      data.j <- ys[zs == j]
      mu.j.params <- nnUpdate(m0,s0, epsilon, data.j)
      mus[j] <- rnorm(1, mean = mu.j.params[1], mu.j.params[2])
    }
    # compute probabilites for each dist of each z_i
    z.probs <- sapply(1:n, function(x) {catProbs(w, mus,
                                              epsilon, ys[x])})
    # sampler z_i
    zs <- apply(z.probs, 2, function(x) {categSampler(x)})
    # store value
    res[i,] <- c(m0,s0,epsilon,mus,w)
  }
  return(res[burnin:n.iter,])
}
```

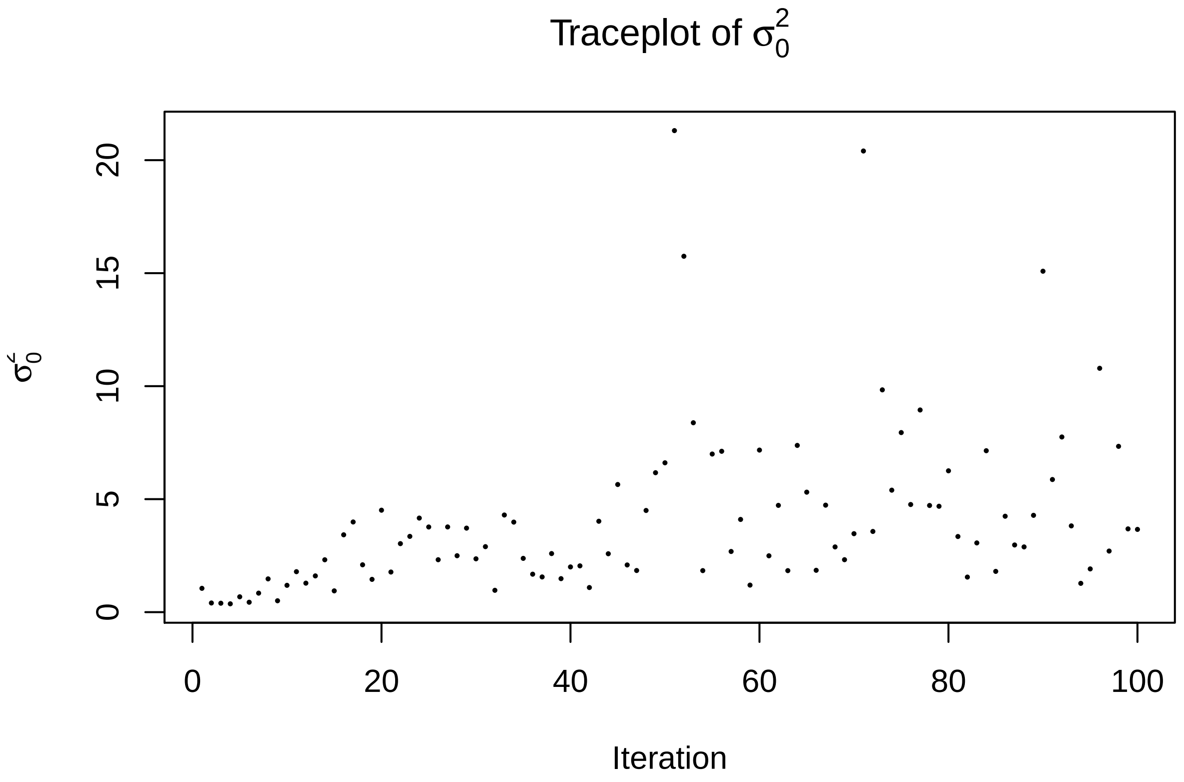
```
# read in data and set parameters
y.data <- read.csv("Lab8Mixture.csv", header = FALSE)$V1
mus <- rnorm(3)
m0 <- 1
s0 <- 1
w <- rdirichlet(1, c(1,1,1))
epsilon <- 5
zs <- replicate(length(y.data), categSampler(w))
n.iter <- 100
```

```
# run sampler
post.samps <- augSampler(y.data, zs, mus, m0, s0, w, epsilon, n.iter)
# name parameters
param.names <- c("$\\mu_0$", "$\\sigma_0^2$","$\\epsilon^2$","$\\mu_1","$\\mu_2$","$\\mu_3$","$w_1$",
```

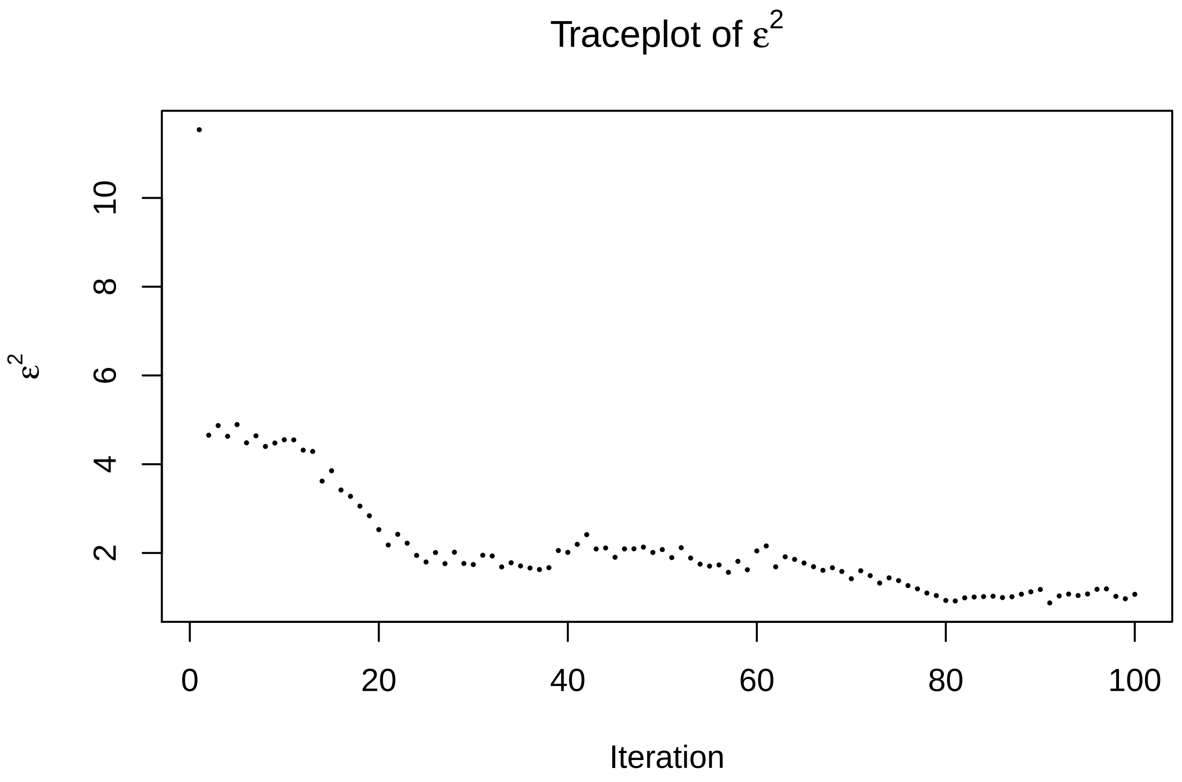
```
plot(1:n.iter, post.samps[,1], pch = 16, cex = .35,
     xlab = "Iteration", ylab = expression(mu[0]),
     main = expression(paste("Traceplot of ", mu[0])))
```



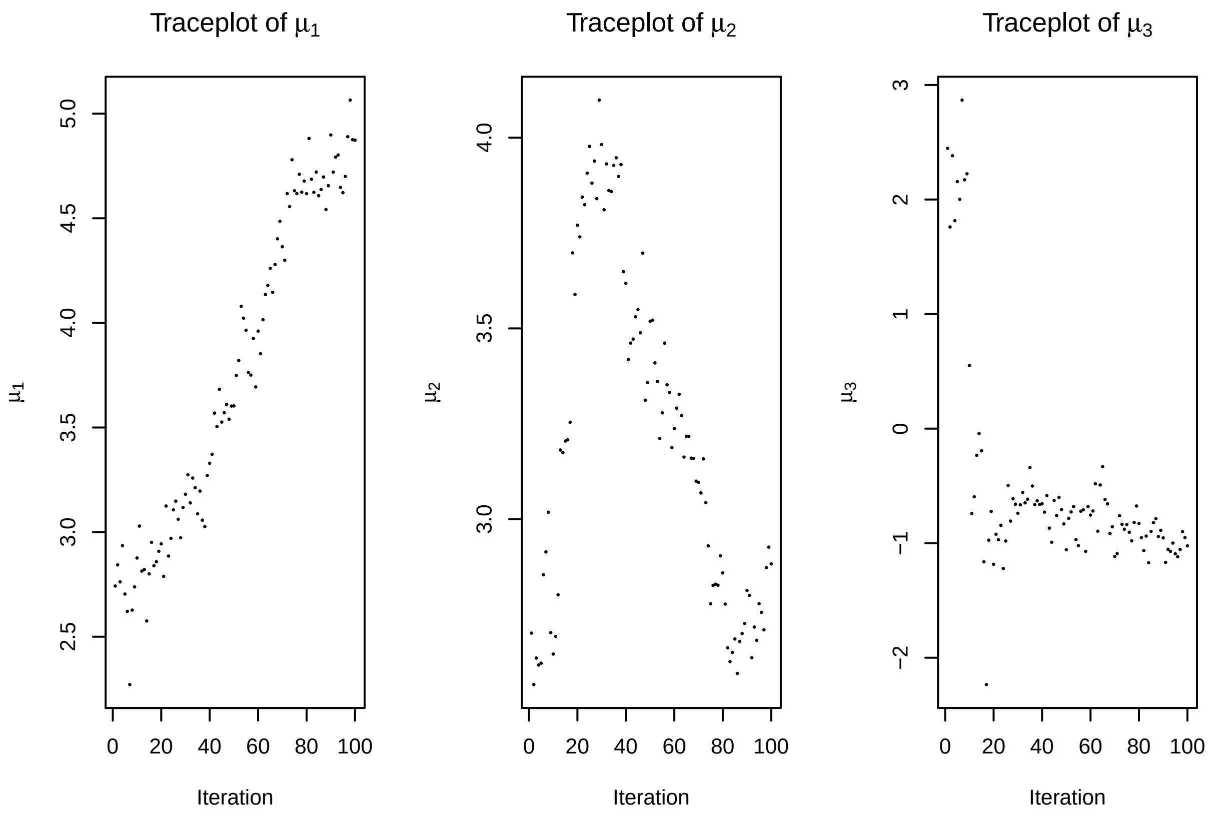
```
plot(1:n.iter, post.samps[,2], pch = 16, cex = .35,
     xlab = "Iteration", ylab = expression(sigma[0]^2),
     main = expression(paste("Traceplot of ", sigma[0]^2)))
```



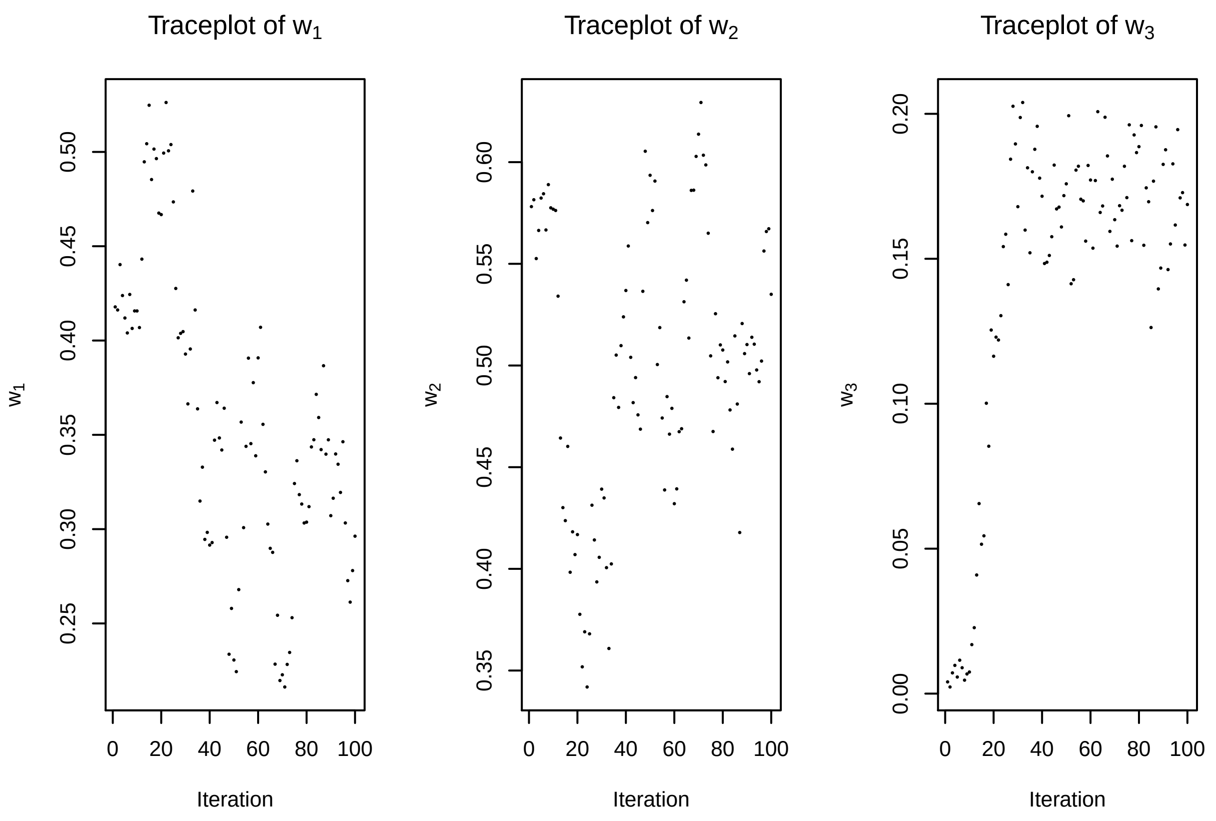
```r
plot(1:n.iter, post.samps[,3], pch = 16, cex = .35,
     xlab = "Iteration", ylab = expression(epsilon^2),
     main = expression(paste("Traceplot of ", epsilon^2)))
```



```r
par(mfrow=c(1,3))
for (ind in 1:3){
  x.lab <- bquote(mu[.(ind)])
  plot(1:n.iter, post.samps[,3+ind], pch = 16, cex = .35,
       xlab = "Iteration", ylab = x.lab,
       main = bquote(paste("Traceplot of ", .(x.lab))))
}
```



```r
par(mfrow=c(1,3))
for (ind in 1:3){
  x.lab <- bquote(w[.(ind)])
  plot(1:n.iter, post.samps[,6+ind], pch = 16, cex = .35,
       xlab = "Iteration", ylab = x.lab,
       main = bquote(paste("Traceplot of ", .(x.lab))))
}
```



```r
# calculate summary statistics and display using xtable
ints <- apply(post.samps, 2, function(x) {quantile(x, c(.025,.975))})
means <- apply(post.samps, 2, mean)
sum.data <- cbind(means, t(ints))
row.names(sum.data) <- param.names
xtable(sum.data, sanitize.colnames.function = identity)
```

```
## % latex table generated in R 3.4.1 by xtable 1.8-2 package
## % Tue Apr  3 14:32:53 2018
## \begin{table}[ht]
## \centering
## \begin{tabular}{rrrr}
##   \hline
##  & means & 2.5\% & 97.5\% \\
##   \hline
## \$$\backslash$mu\_0\$ & 1.66 & -0.43 & 3.09 \\
##   \$$\backslash$sigma\_0\verb|^|2\$ & 4.07 & 0.42 & 15.43 \\
##   \$$\backslash$epsilon\verb|^|2\$ & 2.17 & 0.95 & 4.77 \\
##   \$$\backslash$mu\_1 & 3.75 & 2.62 & 4.89 \\
##   \$$\backslash$mu\_2\$ & 3.21 & 2.62 & 3.96 \\
##   \$$\backslash$mu\_3\$ & -0.53 & -1.18 & 2.31 \\
##   \$w\_1\$ & 0.36 & 0.22 & 0.50 \\
##   \$w\_2\$ & 0.50 & 0.36 & 0.60 \\
##   \$w\_3\$ & 0.14 & 0.01 & 0.20 \\
##    \hline
## \end{tabular}
## \end{table}
```