# HW 3

Nathan Hawkins

2/5/2021

```r
twogroups <- read.table("C:/Users/nateh/Downloads/twogroups.dat")

names(twogroups) <- c("tmt", "y")
```

## 1 Jags Model

```r
library(R2jags)

hw3mdl <- "
  model {

    for(i in 1:33){
      y[i] ~ dnorm(mu[tmt[i]], 1/s2[tmt[i]])
    }

    #Priors
    for(i in 1:2){
      mu[i] ~ dnorm(125, .01)
      s2[i] ~ dunif(0, 2000)
    }


  }
"

writeLines(hw3mdl, 'hw3.txt')
y <- twogroups$y
tmt <- twogroups$tmt
data.jags <- c('y','tmt')
parms <- c("mu", "s2")

hw3.sim <- jags(data = data.jags, inits = NULL,
               parameters.to.save = parms,
               model.file = 'hw3.txt',
               n.iter = 16000,
               n.burnin = 1000,
               n.chains = 4,
               n.thin = 3)

## module glm loaded
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 33
##    Unobserved stochastic nodes: 4
##    Total graph size: 77
##
## Initializing model
```

```
hw3.sim
```

```
## Inference for Bugs model at "hw3.txt", fit using jags,
##  4 chains, each with 16000 iterations (first 1000 discarded), n.thin = 3
##  n.sims = 20000 iterations saved
##          mu.vect sd.vect    2.5%     25%     50%     75%    97.5%  Rhat
n.eff
## mu[1]    131.907   5.331 121.204 128.435 131.996 135.527  142.073 1.001
7900
## mu[2]    121.566   2.070 117.451 120.253 121.549 122.881  125.720 1.001
8100
## s2[1]    767.008 282.990 380.449 564.957 708.323 905.081 1497.371 1.001
20000
## s2[2]     62.822  33.185  26.178  41.380  54.701  74.542  147.215 1.001
5200
## deviance 271.225   3.371 266.938 268.702 270.506 272.918  279.641 1.001
5600
##
## 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 = 5.7 and DIC = 276.9
## DIC is an estimate of expected predictive error (lower deviance is
better).
```

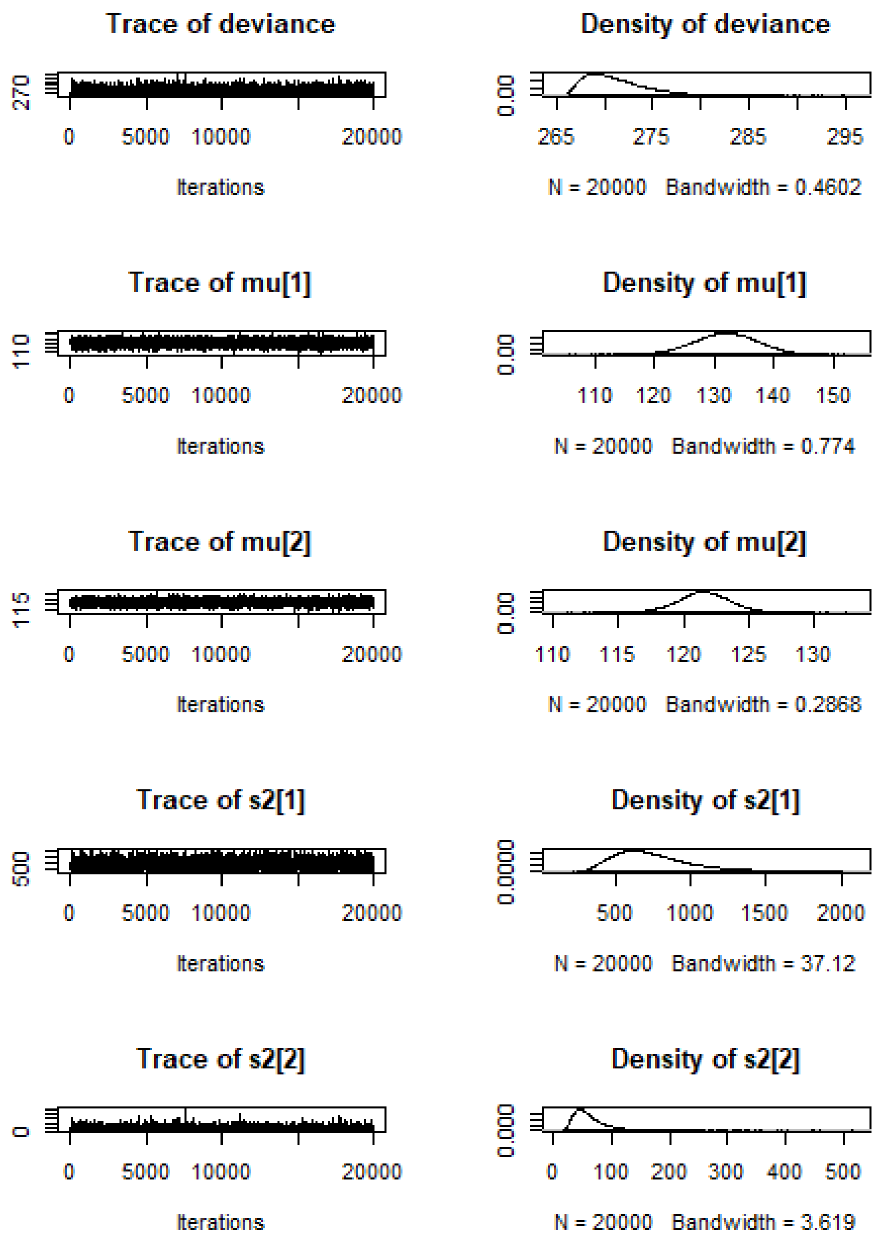
## Convergence checks

The plots look like they converge. Dependence is close to 1. Effective size is plenty high, and there's low autocorrelation.

```
sims <- as.mcmc(hw3.sim)
chains <- as.matrix(sims)
sims <- as.mcmc(chains)
head(sims)
```

```
## Markov Chain Monte Carlo (MCMC) output:
## Start = 1
## End = 7
## Thinning interval = 1
##      deviance   mu[1]   mu[2]   s2[1]    s2[2]
## [1,] 271.3735 138.2258 118.2706 784.4035 73.26400
## [2,] 266.6953 134.0042 120.5847 623.1723 37.02006
## [3,] 271.1994 133.3082 118.2709 905.4388 57.70077
## [4,] 266.8827 134.9346 121.6499 466.0605 44.63889
## [5,] 268.3093 131.8496 121.2842 561.6071 69.36466
## [6,] 271.3003 138.7068 124.4653 512.9549 79.65210
## [7,] 268.5313 136.1771 121.1794 602.4863 74.01456
```

```r
plot(sims, type = 'l')
```



```
mu1 <- sims[,2]
mu2 <- sims[,3]
s2_1 <- sims[,4]
s2_2 <- sims[,5]
```

```
effectiveSize(sims)

##  deviance     mu[1]     mu[2]     s2[1]     s2[2] 
## 12530.547 19031.507 20333.903 15756.649  9711.661

autocorr.diag(sims)

##             deviance          mu[1]        mu[2]        s2[1]         s2[2]
## Lag 0   1.0000000000  1.0000000000  1.000000000  1.000000000  1.0000000000
## Lag 1   0.1880042909  0.0042503403 -0.001850868  0.118648420  0.2930639307
## Lag 5   0.0009060010  0.0037843733 -0.005122316 -0.002792146  0.0009557271
## Lag 10 -0.0019990740 -0.0003755464  0.012272583 -0.009289859 -0.0011706081
## Lag 50  0.0007688069  0.0012880644 -0.003235746 -0.003075261  0.0057581114

raftery.diag(sims)

## 
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95 
## 
##           Burn-in  Total Lower bound  Dependence
##           (M)      (N)   (Nmin)       factor (I)
##  deviance 2        3897  3746         1.040     
##  mu[1]    2        3740  3746         0.998     
##  mu[2]    2        3834  3746         1.020     
##  s2[1]    2        3665  3746         0.978     
##  s2[2]    2        3695  3746         0.986
```

## Next Model

```
library(R2jags)
a2mdl <- "
  model {

    for(i in 1:33){
      y[i] ~ dnorm(mu[tmt[i]], 1/s2[tmt[i]])
    }

    #Priors
    for(i in 1:2){
      mu[i] ~ dnorm(125, .01)
    }

    s2[1] ~ dgamma(2, .002)
    s2[2] ~ dgamma(2, .04)


  }
```

```
"

writeLines(a2mdl, 'a1.txt')
y <- twogroups$y
tmt <- twogroups$tmt
data.jags <- c('y','tmt')
parms <- c("mu", "s2")

a1.sim <- jags(data = data.jags, inits = NULL,
               parameters.to.save = parms,
               model.file = 'a1.txt',
               n.iter = 16000,
               n.burnin = 1000,
               n.chains = 4,
               n.thin = 3)

## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 33
##    Unobserved stochastic nodes: 4
##    Total graph size: 78
##
## Initializing model

a1.sim

## Inference for Bugs model at "a1.txt", fit using jags,
##  4 chains, each with 16000 iterations (first 1000 discarded), n.thin = 3
##  n.sims = 20000 iterations saved
##          mu.vect sd.vect    2.5%     25%     50%     75%    97.5%  Rhat
n.eff
## mu[1]    132.040   5.269 121.423 128.571 132.142 135.580  142.255 1.001
20000
## mu[2]    121.542   1.827 117.927 120.353 121.526 122.718  125.149 1.001
18000
## s2[1]    718.277 245.852 377.738 545.816 672.488 841.174 1319.428 1.001
20000
## s2[2]     49.344  18.329  24.151  36.246  45.867  58.505   94.436 1.001
20000
## deviance 270.138   2.616 266.779 268.214 269.575 271.465  276.823 1.001
9800
##
## 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 = 3.4 and DIC = 273.6
## DIC is an estimate of expected predictive error (lower deviance is
better).
```

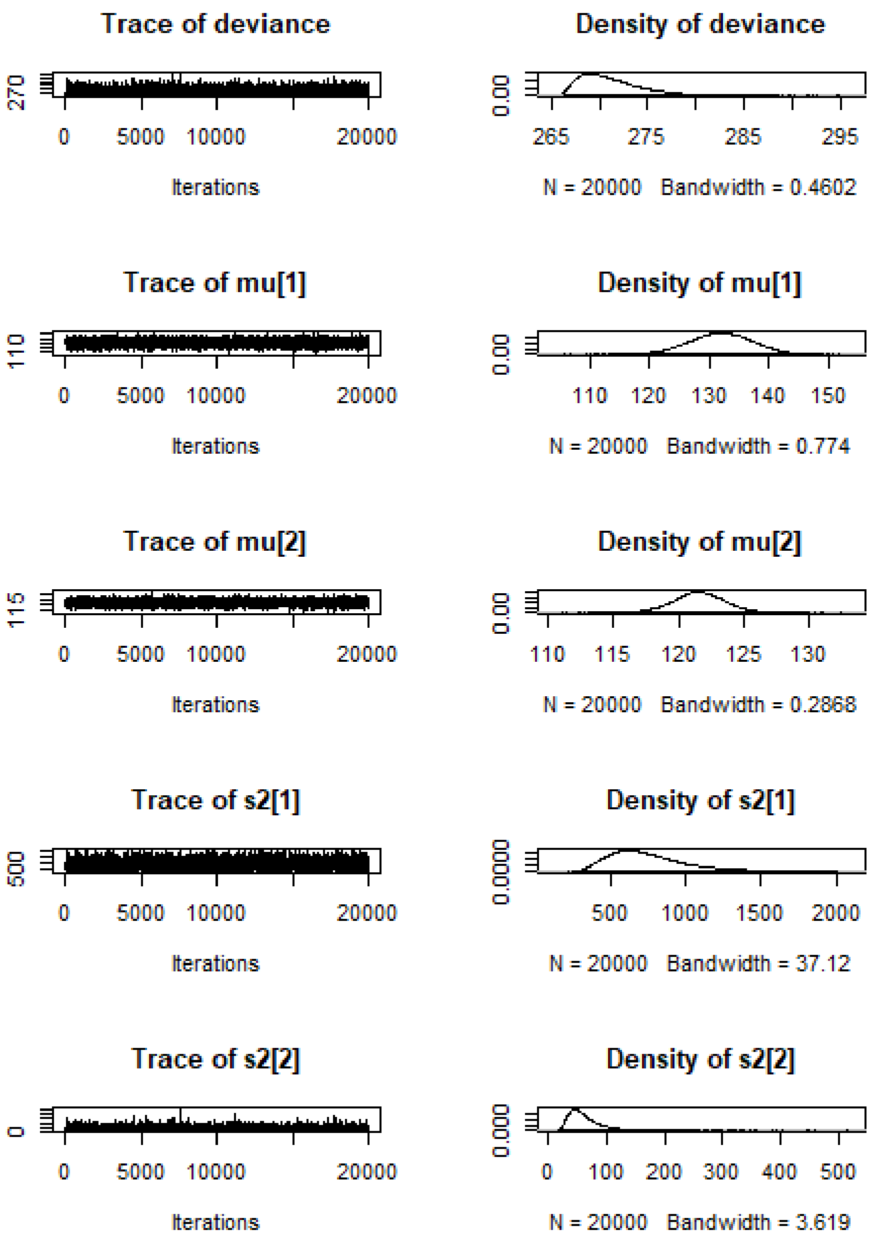
## Convergence checks

The plots look like they converge. Dependence is close to 1. Effective size is plenty high, and there's low autocorrelation.

```
a1_sims <- as.mcmc(a1.sim)
a1_chains <- as.matrix(a1_sims)
a1_sims <- as.mcmc(a1_chains)
head(a1_sims)
```

```
## Markov Chain Monte Carlo (MCMC) output:
## Start = 1
## End = 7
## Thinning interval = 1
##      deviance    mu[1]    mu[2]     s2[1]    s2[2]
## [1,] 270.6537 138.4488 124.0462  929.4059 44.30426
## [2,] 274.7284 119.9144 120.5744  845.6950 24.89042
## [3,] 269.9547 133.6447 121.0906  553.4022 90.84452
## [4,] 267.6281 133.8651 120.5717  731.4693 31.29615
## [5,] 269.8401 127.6098 119.4809  837.4337 45.98096
## [6,] 272.7338 132.1013 120.6334 1146.1108 23.73583
## [7,] 276.7017 131.9503 126.8763  742.0794 43.08941
```

```
plot(sims, type = 'l')
```



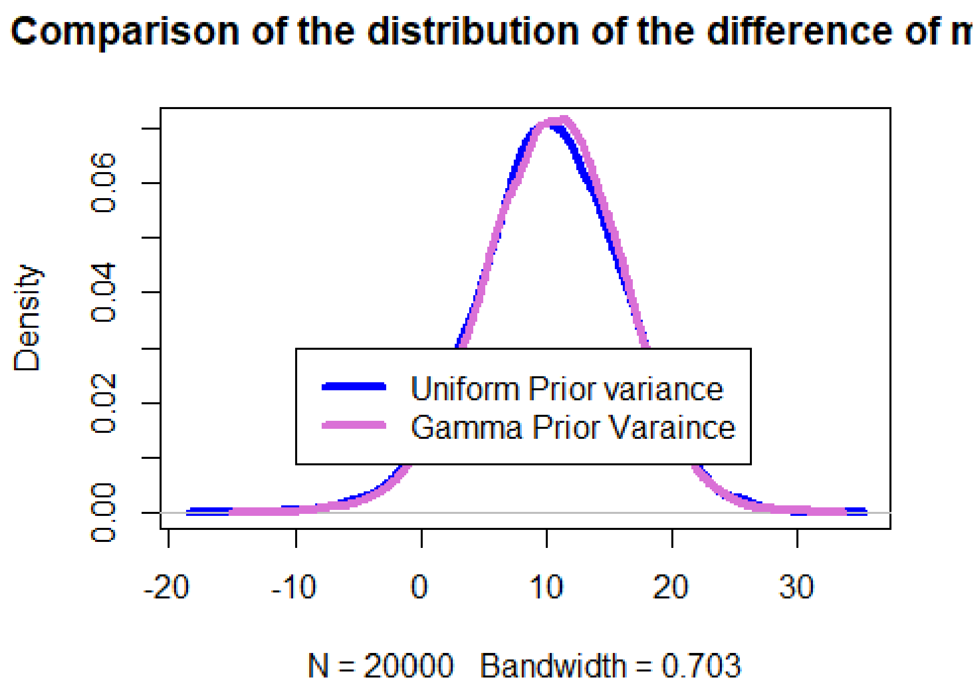
```
a1_mu1 <- a1_sims[,2]
a1_mu2 <- a1_sims[,3]
a1_s2_1 <- a1_sims[,4]
a1_s2_2 <- a1_sims[,5]
```

```
#Effective Size
effectiveSize(a1_sims)

## deviance    mu[1]    mu[2]    s2[1]    s2[2]
## 16336.95 20000.00 20000.00 15806.15 14766.63

#Autocorrelation
autocorr.diag(a1_sims)

##            deviance        mu[1]         mu[2]        s2[1]        s2[2]
## Lag 0  1.000000e+00  1.000000000  1.0000000000  1.000000000  1.000000000
## Lag 1  7.987511e-02 -0.003975177 -0.0038395571  0.117101959  0.137804593
## Lag 5  8.045588e-04  0.001839992  0.0074255444 -0.005034121 -0.004102734
## Lag 10 6.194174e-05  0.001936787  0.0004039943 -0.004916510  0.008478139
## Lag 50 2.322618e-03 -0.003780772  0.0042630126  0.001396114 -0.013552910

#Diagnositcs
raftery.diag(a1_sims)

## 
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95 
## 
##           Burn-in  Total Lower bound  Dependence
##           (M)      (N)   (Nmin)       factor (I)
##  deviance 2        3680  3746         0.982     
##  mu[1]    2        3710  3746         0.990     
##  mu[2]    1        3755  3746         1.000     
##  s2[1]    2        3771  3746         1.010     
##  s2[2]    2        3787  3746         1.010
```

## Comparing Models

DIC is lower for the model with the gamma variances. Here are density plots of the differences

```
diff1 <- mu1-mu2
diff2 <- a1_mu1 - a1_mu2
plot(density(diff1), col = 'blue', lwd = 4, main = "Comparison of the
distribution of the difference of mu's")
lines(density(diff2), col = 'orchid', lwd = 4)
legend(-10, .03, c("Uniform Prior variance", "Gamma Prior Varaince"), col =
c("blue", "orchid"), lty = 1, lwd = 4)
```



# 2 Same thing but in sas Sas Code

library(SASmarkdown)

saspath <- "C:/Program Files/SASHome/SASFoundation/9.4/sas.exe" sasopts <- "-nosplash -ls 75" knitr::opts_chunk$set(engine="sas", engine.path=saspath, engine.opts=sasopts, comment=NA)

knitr::opts_chunk$get()$engine knitr::opts_chunk$get()$engine.path knitr::opts_chunk$get()$engine.opts

## Sas Code. Works in R just not when I knit it.

data twogroups; infile 'C:/Users/nateh/Downloads/twogroups.dat'; input tmt y; run;

proc means data=twogroups; run;

proc mcmc data = twogroups nbi = 30000 nmc = 300000 thin = 30 outpost = 'C:/Users/nateh/Documents/Stat 451/anovamodel_1.sas7bdat' dic propcov=quanew monitor=($parms$) stats = all diagnostics = all;

array mu[2] mu1-mu2;

array vv[2] vv1-vv2;

parms mu1: 0;

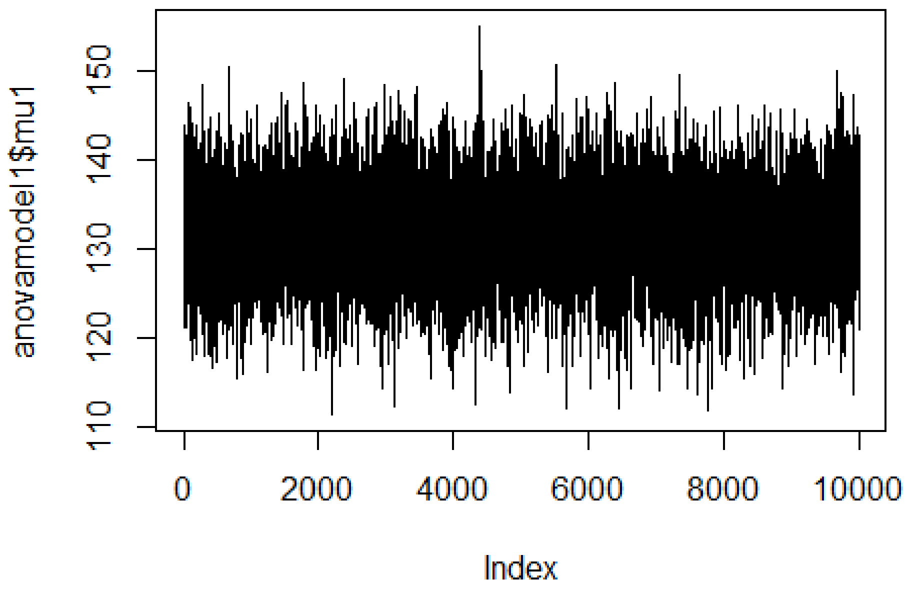
```
parms mu2: 0;

 parms vv1: 10;

parms vv2: 10;

prior mu1: ~ normal(125, var = 100);

prior mu2: ~ normal(125, var = 100);

prior vv1: ~ uniform(0, 2000);

prior vv2: ~ uniform(0, 2000);

 model y ~ normal(mu[tmt], var = vv[tmt]);

run;
```

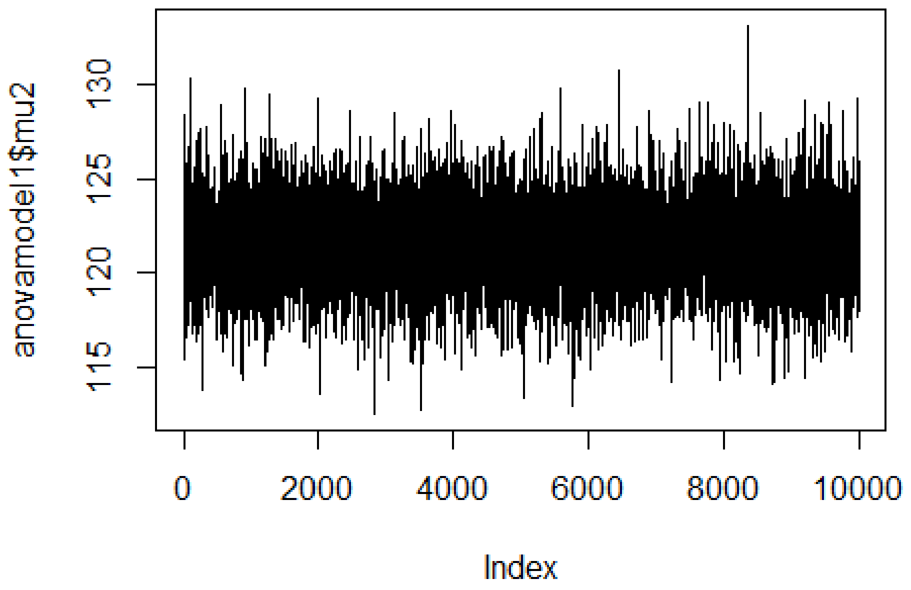
2nd model

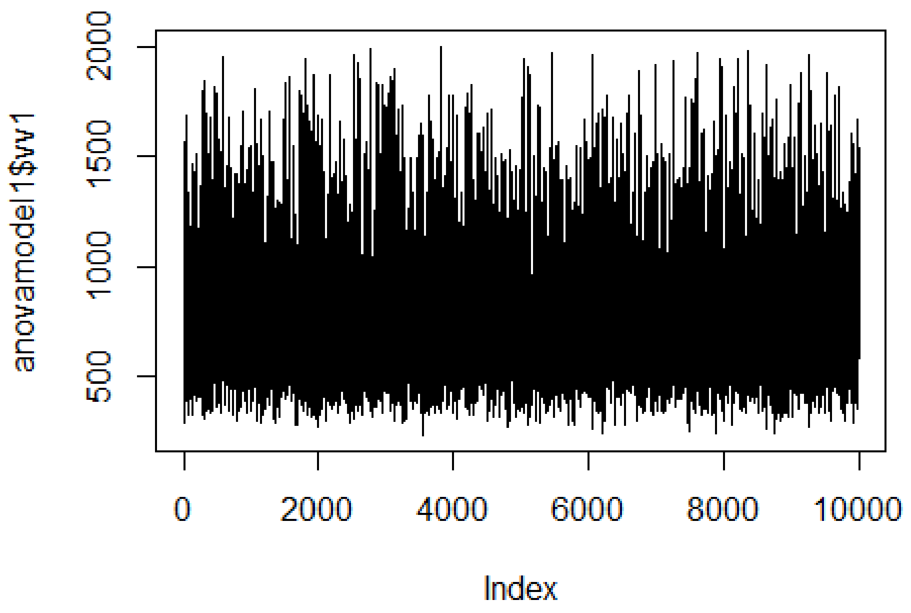
```
data twogroups; infile ‘C:/Users/nateh/Downloads/twogroups.dat’; input tmt y; run;

proc means data=twogroups; run;

proc mcmc data = twogroups nbi = 30000 nmc = 300000 thin = 30 outpost =
‘C:/Users/nateh/Documents/Stat 451/anovamodel_2.sas7bdat’ dic propcov=quanew
monitor=(parms) stats = all diagnostics = all; array mu[2] mu1-mu2; array vv[2] vv1-vv2;
parms mu1: 0; parms mu2: 0; parms vv1: 10; parms vv2: 10; prior mu1: ~ normal(125, var
= 100); prior mu2: ~ normal(125, var = 100); prior vv1: ~ gamma(2, scale = 500); prior
vv2: ~ gamma(2, scale = 25); model y ~ normal(mu[tmt], var = vv[tmt]); run; endsas;
```

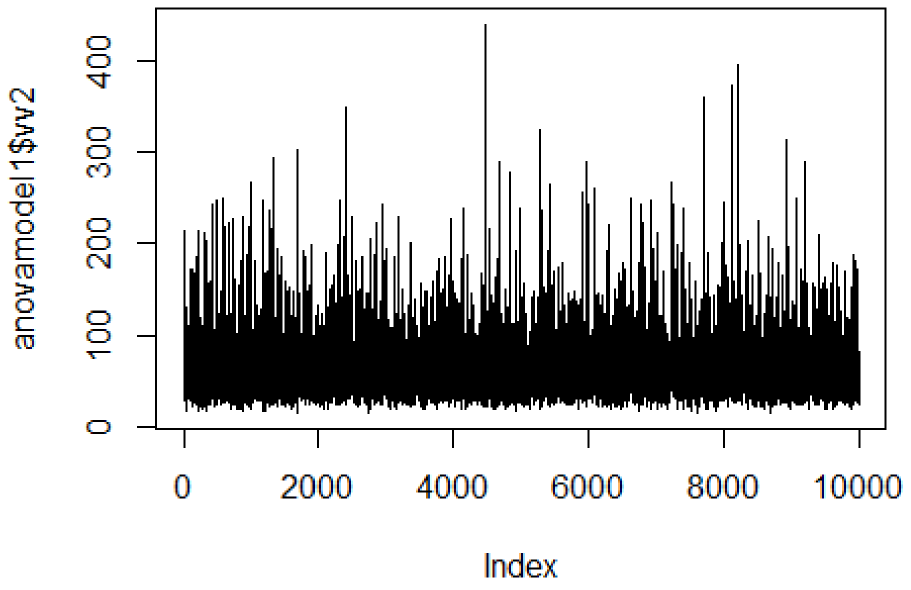
```
# Now back to R, Convergence Checks
```

#Check Convergence for sas model









```
## Markov Chain Monte Carlo (MCMC) output:
## Start = 1
## End = 7
## Thinning interval = 1
##      Iteration      mu1      mu2       vv1       vv2  LogPrior   LogLike
## [1,]     30001 143.8716 121.6742  912.9847  33.46886 -23.48085 -135.0361
## [2,]     30031 135.6311 118.0768  552.6784  81.41652 -22.44960 -135.5289
## [3,]     30061 136.2304 124.0955  827.1403  50.95454 -22.27956 -134.8728
## [4,]     30091 129.9768 120.9021  741.7832  51.25270 -21.85266 -133.9214
## [5,]     30121 125.7824 115.3936  652.9950  89.26981 -22.10933 -138.9344
## [6,]     30151 121.1100 120.1923  666.5678 101.96143 -21.83608 -138.2100
## [7,]     30181 129.3298 128.4547 1028.0689 214.86834 -21.79826 -142.3405
##        LogPost
## [1,] -158.5170
## [2,] -157.9785
## [3,] -157.1524
## [4,] -155.7741
## [5,] -161.0437
## [6,] -160.0460
## [7,] -164.1387
```

```
## Iteration       mu1       mu2       vv1       vv2  LogPrior   LogLike
LogPost
##      0.00  10287.70  10000.00  10000.00  10000.00  11436.64  10000.00
10002.44
```

```
##        Iteration          mu1          mu2         vv1          vv2
## Lag 0  1.0000000  1.000000000  1.000000000 1.000000000  1.000000000
## Lag 1  0.9997000 -0.014231214 -0.007914044 0.008984984  0.008529609
## Lag 5  0.9985000 -0.003150813  0.007266876 0.001238803 -0.003675349
## Lag 10 0.9970000 -0.007077288  0.005002860 0.002367592  0.005092706
## Lag 50 0.9850002  0.015948987  0.007268022 0.002873098  0.013671493
##             LogPrior      LogLike      LogPost
## Lag 0   1.0000000000  1.000000000  1.000000000
## Lag 1  -0.0057868945  0.007821569  0.012344691
## Lag 5  -0.0027864119  0.001129514 -0.002953650
## Lag 10 -0.0007150263 -0.006340446 -0.005898775
## Lag 50  0.0069705491  0.017121506  0.011912975
```

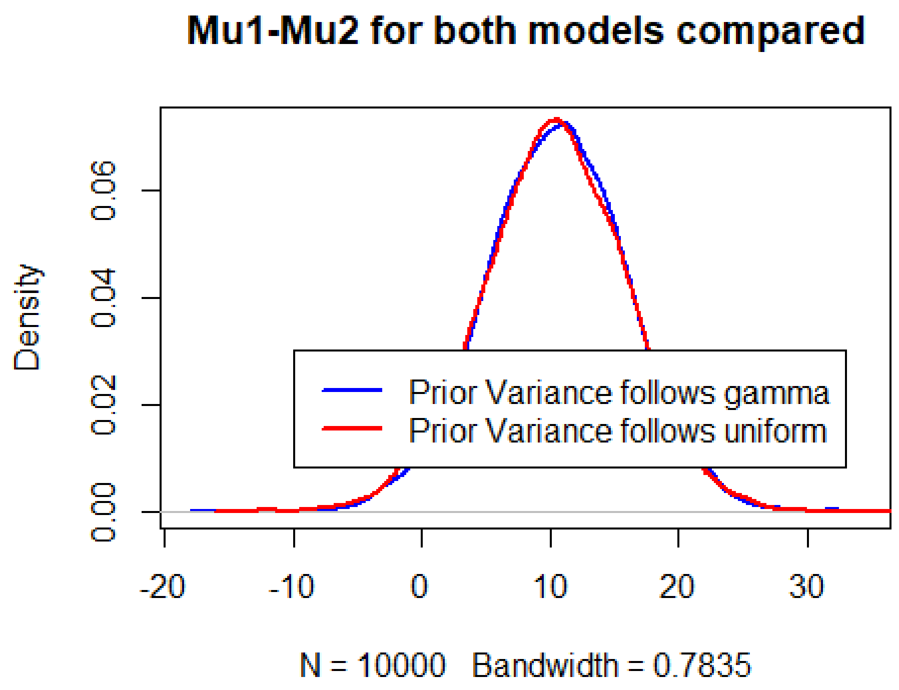
```
## 
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95 
## 
##            Burn-in  Total Lower bound  Dependence
##            (M)      (N)   (Nmin)       factor (I)
##  Iteration 1724     1724  3746         0.460     
##  mu1       2        3620  3746         0.966     
##  mu2       2        3710  3746         0.990     
##  vv1       2        3771  3746         1.010     
##  vv2       2        3620  3746         0.966     
##  LogPrior  2        3851  3746         1.030     
##  LogLike   2        3650  3746         0.974     
##  LogPost   2        3680  3746         0.982
## Markov Chain Monte Carlo (MCMC) output:
## Start = 1
## End = 7
## Thinning interval = 1
##      Iteration      mu1      mu2      vv1      vv2  LogPrior   LogLike
## [1,]     30001 137.8225 123.3394 529.7137 61.00361 -19.26219 -134.3175
## [2,]     30031 125.5393 120.1189 939.4121 30.87766 -18.26924 -135.6062
## [3,]     30061 124.1222 119.9756 537.2801 52.50988 -18.36752 -135.6387
## [4,]     30091 132.2810 121.5039 551.4720 36.11525 -18.28444 -133.3087
## [5,]     30121 128.2311 118.5680 508.9173 61.01181 -18.68402 -135.4639
## [6,]     30151 137.2801 122.9064 462.5257 61.95552 -19.22608 -134.3651
## [7,]     30181 118.8149 116.2278 973.5619 62.81285 -19.32456 -140.2862
##        LogPost
## [1,] -153.5797
## [2,] -153.8754
## [3,] -154.0062
## [4,] -151.5932
## [5,] -154.1480
## [6,] -153.5911
## [7,] -159.6108
```

```
## Iteration       mu1       mu2       vv1       vv2  LogPrior   LogLike 
LogPost 
##     0.000 10000.000 10000.000 10000.000  9602.729  9712.946 10000.000 
10000.000
```

```
##        Iteration           mu1          mu2          vv1           vv2
## Lag 0  1.0000000  1.0000000000  1.000000000  1.000000000  1.0000000000
## Lag 1  0.9997000  0.0085170394 -0.010621960  0.002426704  0.0202161217
## Lag 5  0.9985000  0.0053861727  0.008536314  0.010974915 -0.0004993105
## Lag 10 0.9970000 -0.0012004054  0.007418443 -0.007817121 -0.0055098663
## Lag 50 0.9850002 -0.0004445261  0.003926898  0.002753726 -0.0013077705
##            LogPrior      LogLike      LogPost
## Lag 0   1.000000000  1.000000000  1.000000000
## Lag 1   0.014511710 -0.007026406  0.001362629
## Lag 5   0.017602756  0.009944434  0.015964661
## Lag 10 -0.012383189  0.002910615 -0.001959919
## Lag 50  0.009533903  0.001255593  0.004555453
```

```
## 
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95 
## 
##            Burn-in  Total Lower bound  Dependence
##            (M)      (N)   (Nmin)       factor (I)
##  Iteration 1724     1724  3746         0.460     
##  mu1       2        3741  3746         0.999     
##  mu2       2        3741  3746         0.999     
##  vv1       2        3710  3746         0.990     
##   vv2        2         3680  3746          0.982
##   LogPrior   2         3802  3746          1.010
##   LogLike    2         3741  3746          0.999
##   LogPost    2         3802  3746          1.010
```

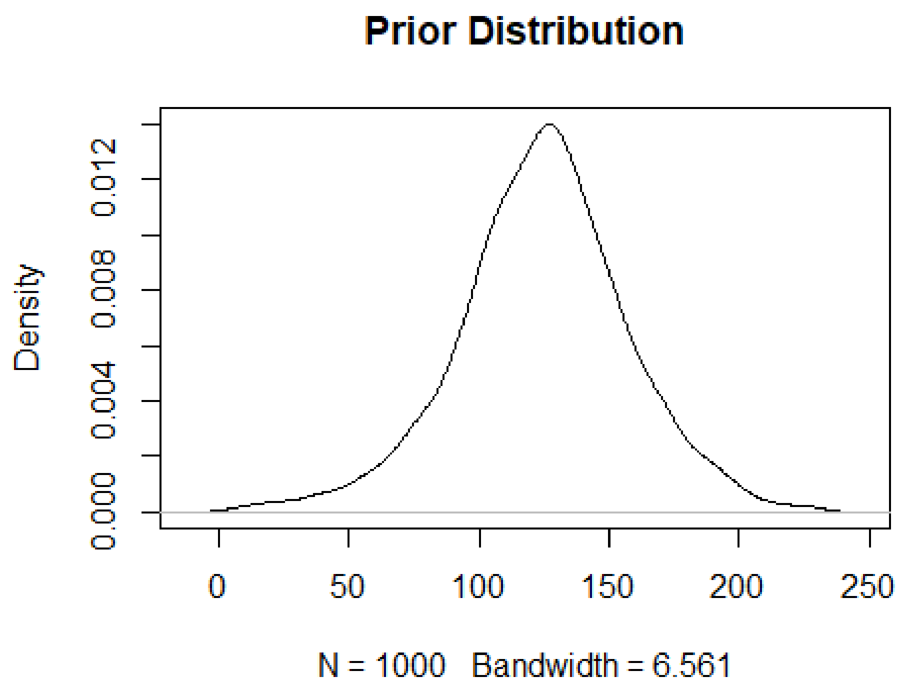
```
plot(density(anovamodel2$mu1 - anovamodel2$mu2), col = 'blue', lwd = 2, main
= "Mu1-Mu2 for both models compared")
lines(density(anovamodel1$mu1 - anovamodel2$mu2), col = 'red', lwd = 2)
legend(-10, .03, c("Prior Variance follows gamma", "Prior Variance follows
uniform"), col = c('blue', 'red'), lty = 1,lwd = 2)
```



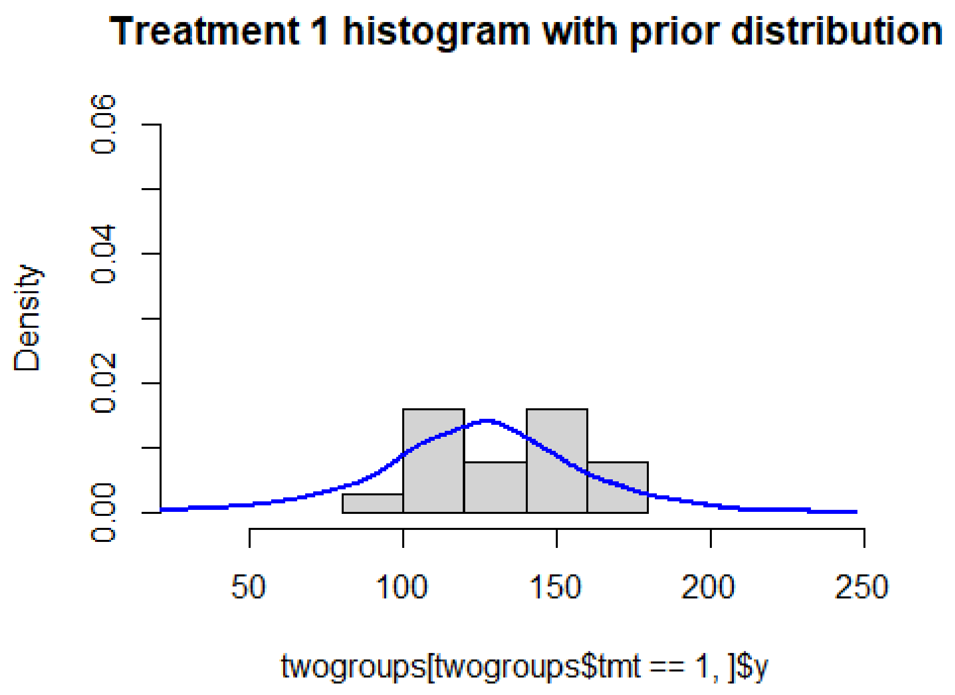
## 3 Prior Predictive for uniform

Prior Predictive Distributions

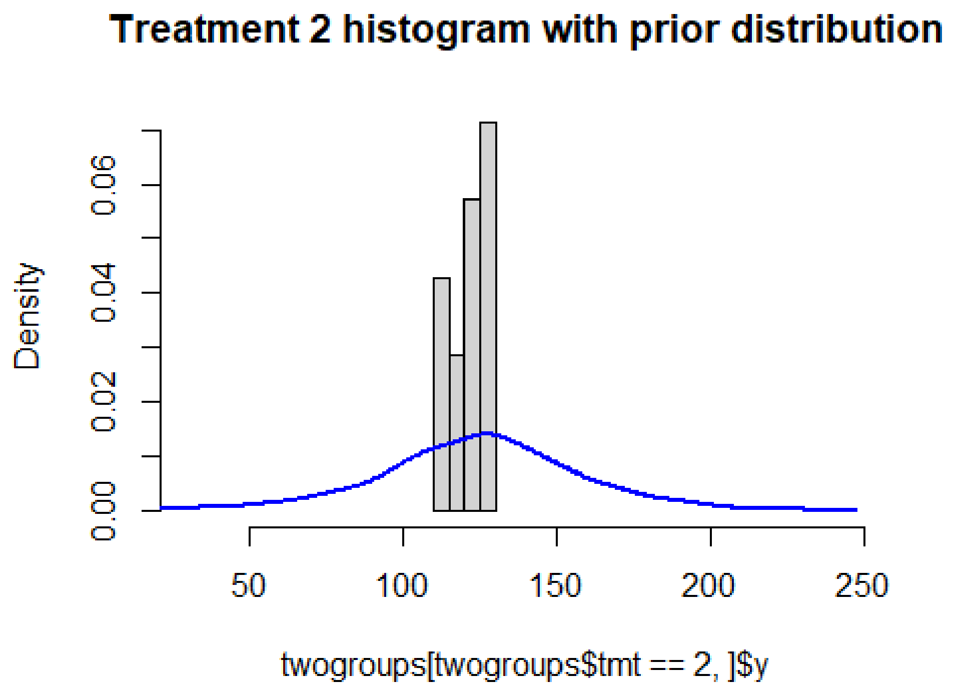
```
# Generate prior distribution
prpred_1 <- NULL
for(i in 1:1000){
  prpred_1[i] <- rnorm(1, 125, sqrt(100)) + rnorm(1,0,sqrt(runif(1,0,2000)))
}
#Plot Prior
plot(density(prpred_1), type = 'l', main = "Prior Distribution")
```



```r
#Plot Prior with treatments
hist(twogroups[twogroups$tmt == 1,]$y, freq =  FALSE, main = "Treatment 1
histogram with prior distribution", xlim = c(30, 250), ylim = c(0, 0.06))
lines(density(prpred_1), type = 'l', col = 'blue', lwd = 2)
```



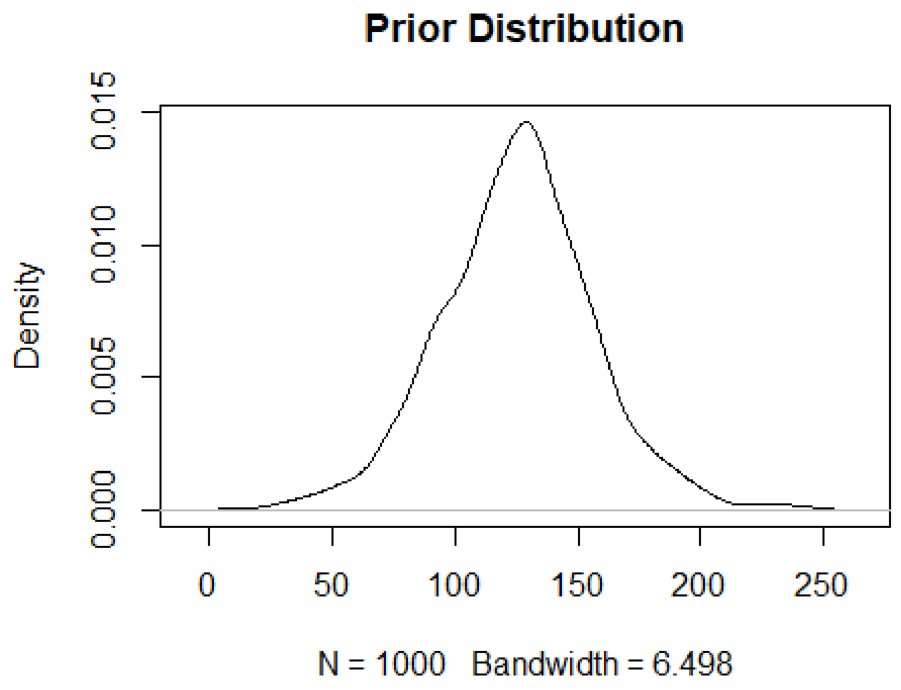
```
hist(twogroups[twogroups$tmt == 2,]$y, freq =  FALSE, main = "Treatment 2
histogram with prior distribution", xlim = c(30, 250))
lines(density(prpred_1), type = 'l', col = 'blue', lwd = 2)
```



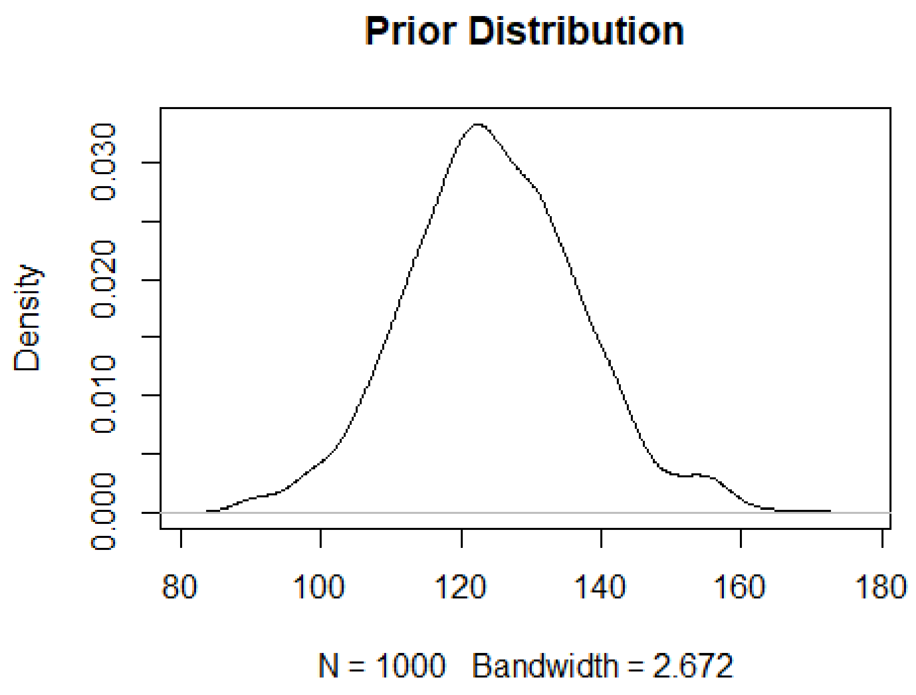
## 4 Prior Predictive for gamma model

Prior Predictive Distributions

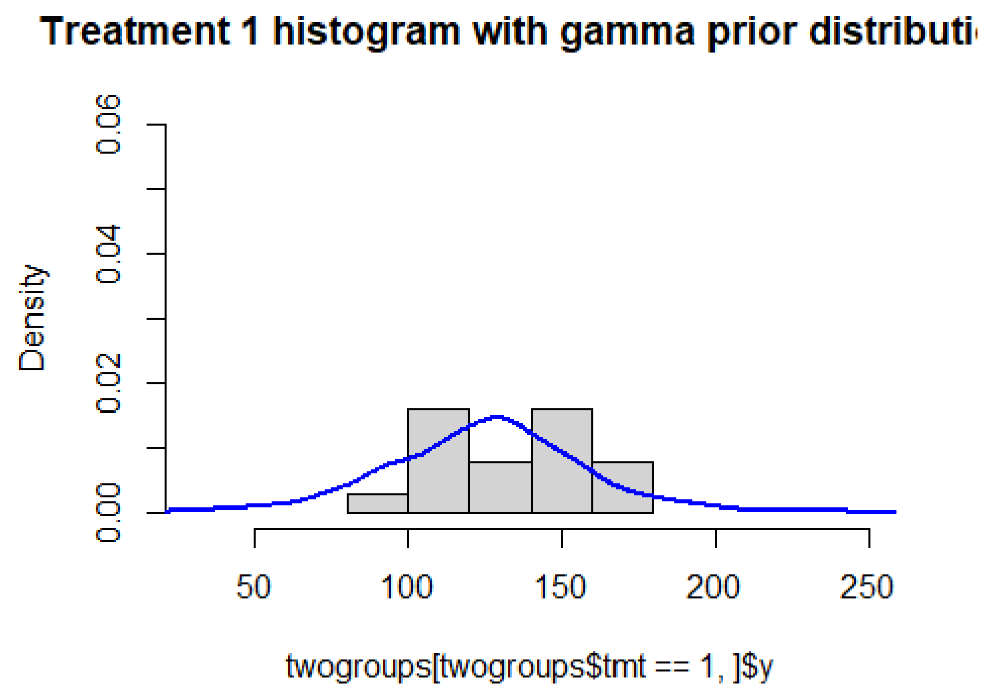
```
# Generate prior distribution
prpred_2 <- NULL
for(i in 1:1000){
  prpred_2[i] <- rnorm(1, 125, sqrt(100)) + rnorm(1,0,sqrt(rgamma(1, 2,
0.002)))
}

prpred_3 <- NULL
for(i in 1:1000){
  prpred_3[i] <- rnorm(1, 125, sqrt(100)) + rnorm(1,0,sqrt(rgamma(1, 2,
0.04)))
}


#Plot Prior
plot(density(prpred_2), type = 'l', main = "Prior Distribution")
```



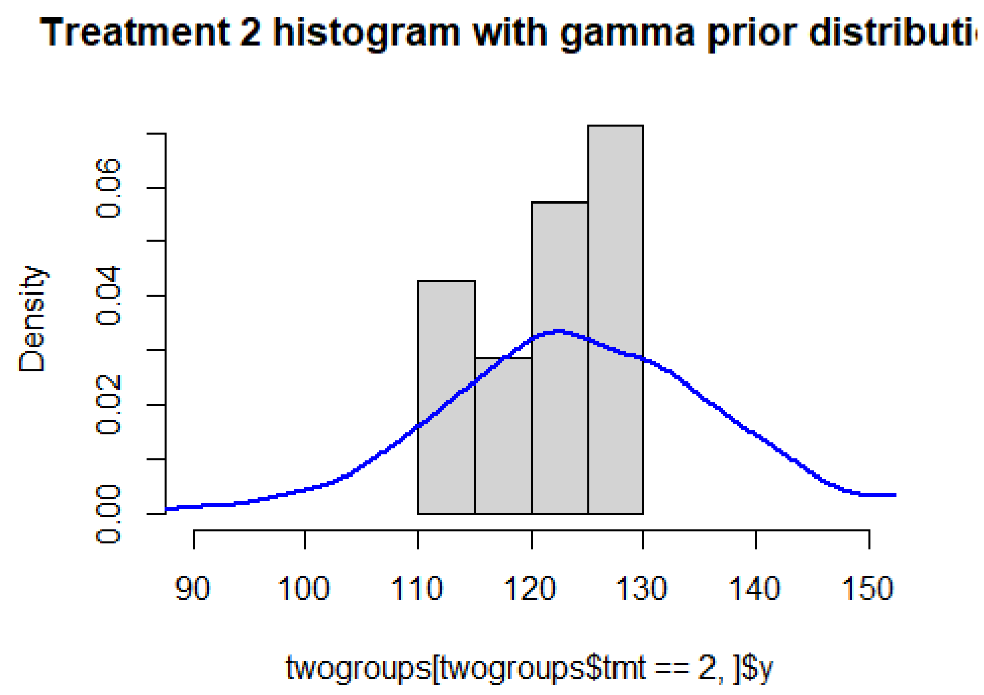
```
plot(density(prpred_3), type = 'l', main = "Prior Distribution")
```



```
#Plot Prior with treatments
hist(twogroups[twogroups$tmt == 1,]$y, freq =  FALSE, main = "Treatment 1
histogram with gamma prior distribution", xlim = c(30, 250), ylim = c(0,
0.06))
lines(density(prpred_2), type = 'l', col = 'blue', lwd = 2)
```



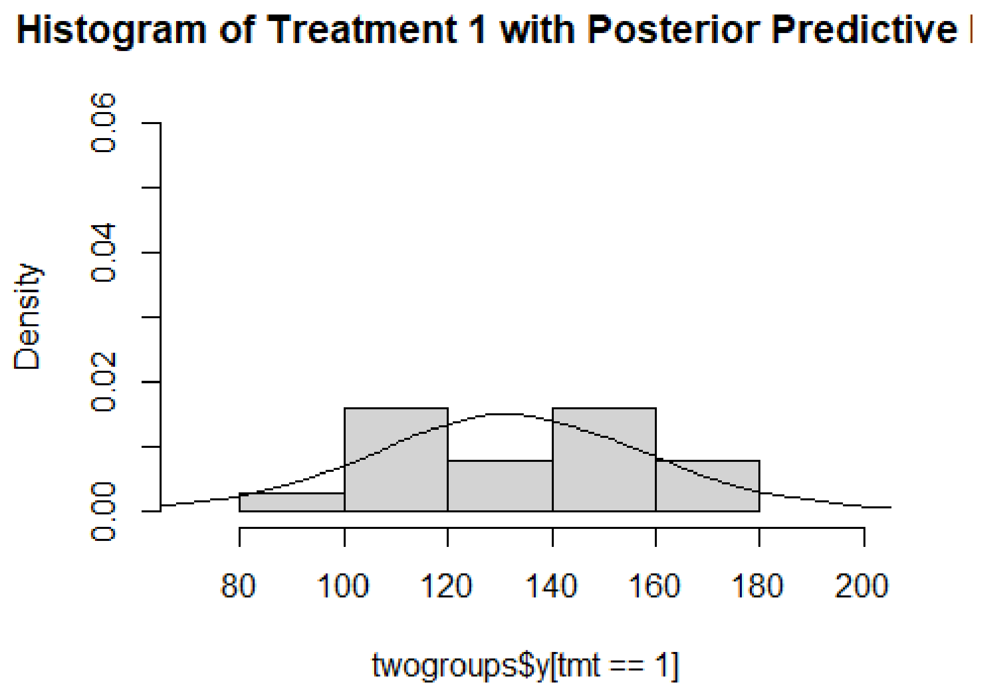
```
hist(twogroups[twogroups$tmt == 2,]$y, freq =  FALSE, main = "Treatment 2
histogram with gamma prior distribution", xlim = c(90, 150))
lines(density(prpred_3), type = 'l', col = 'blue', lwd = 2)
```



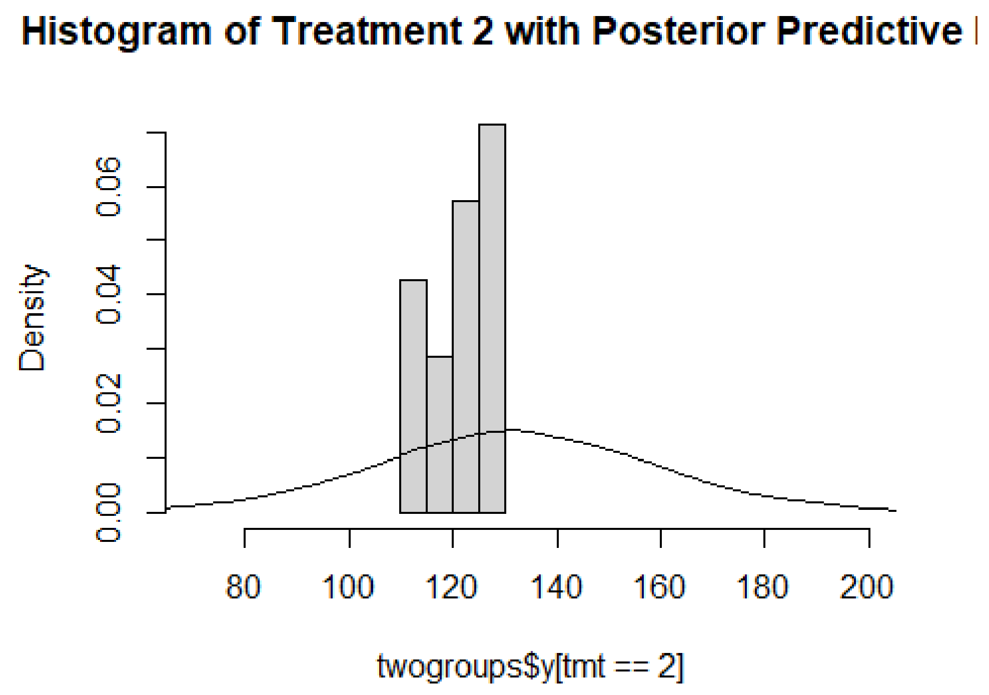
## 5 Posterior Predictive from Jags model

Posterior Predictive Distributions

```
#Posterior Predictive
popdat1 <- NULL
for(i in 1:20000){
            #Noise in mean, #Noise in Error
  popdat1[i] <- mu1[i] + rnorm(1,0,sqrt(s2_1[i]))
}

hist(twogroups$y[tmt == 1], breaks = 5, freq = FALSE, xlim = c(70, 200), main
= "Histogram of Treatment 1 with Posterior Predictive line", ylim = c(0,
0.06))
lines(density(popdat1))
```



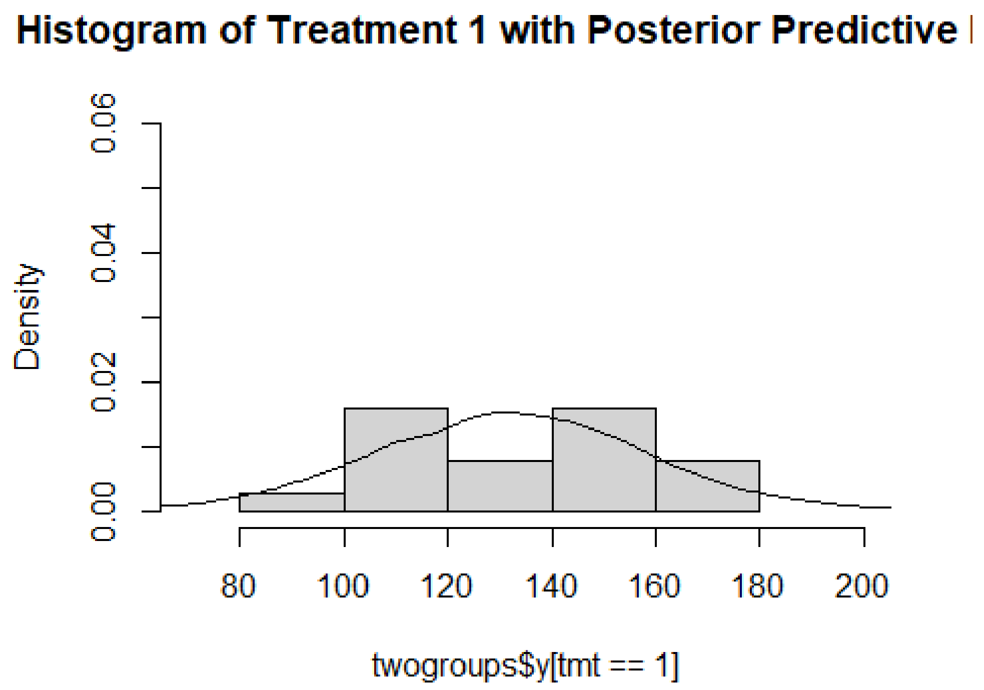
```
hist(twogroups$y[tmt == 2], breaks = 5, freq = FALSE, xlim = c(70, 200), main
= "Histogram of Treatment 2 with Posterior Predictive line")
lines(density(popdat1))
```



## 6 Posterior Predictive for SAS model

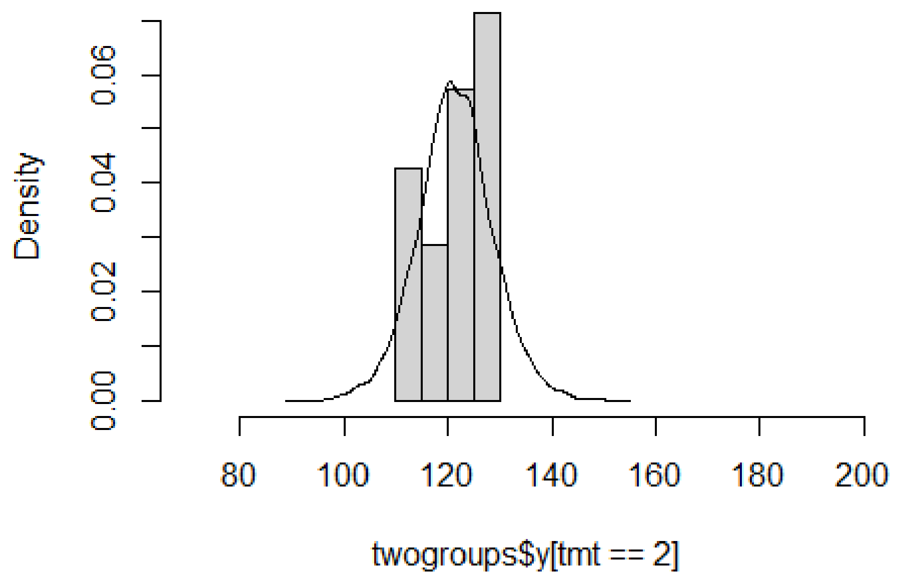
Posterior Predictive Distributions for the 2nd model. These posterior predictive lines closely follow the histogram

```
#Posterior Predictive
sas_mu1 <- an_2.sims[,2]
sas_mu2 <- an_2.sims[,3]
sas_vv1 <- an_2.sims[,4]
sas_vv2 <- an_2.sims[,5]

popdat2_sas <- NULL

for(i in 1:10000){
            #Noise in mean, #Noise in Error
  popdat2_sas[i] <- sas_mu1[i] + rnorm(1,0,sqrt(sas_vv1[i]))
}

popdat3_sas <- NULL

for(i in 1:10000){
  popdat3_sas[i] <- sas_mu2[i] + rnorm(1,0,sqrt(sas_vv2[i]))
}

hist(twogroups$y[tmt == 1], breaks = 5, freq = FALSE, xlim = c(70, 200), main
= "Histogram of Treatment 1 with Posterior Predictive line", ylim = c(0,
0.06))
lines(density(popdat2_sas))
```



```
hist(twogroups$y[tmt == 2], breaks = 5, freq = FALSE, xlim = c(70, 200), main
= "Histogram of Treatment 2 with Posterior Predictive line")
lines(density(popdat3_sas))
```

**Histogram of Treatment 2 with Posterior Predictive**


Density
twogroups$y[tmt == 2]