BDA - Assignment 7 (part 1)

Anonymous

31 10 2019

Contents

Used libraries:

```
library(dplyr)
library(ggplot2)
library(rstan)
library(gdata)
library(bayesplot)
library(aaltobda)
data("drowning")
```

```
options(mc.cores = parallel::detectCores())
rstan_options(auto_write = TRUE)
```

1 Linear model: drowning data with Stan.

1. Spotting two crucial mistakes

NOTE: I will show the full stan code in the latter parts, as I implement the prior. For now, I only point out the problems.

Original code:

```
#data {
#  int<lower=0> N;
#  vector[N] x;
#  vector[N] y;
#  real xpred;
#}
#parameters {
#  real alpha;
#  real beta;
#  real<upper=0> sigma;   HEP! This is problem 1.
#}
#transformed parameters {
#  vector[N] mu;
#  mu = alpha + beta*x;
#}
#model {
#  y ~ normal(mu, sigma);
#}
#generated quantities {
#  real ypred;
#  ypred = normal_rng(mu, sigma);  HEP! This is problem 2.
#}
```

So, **problem 1** is giving an upper bound of zero for deviance (this would result in trying to poll from a distribution with a negative variance). This should be changed to lower bound (parameters block).

Problem 2. is using mu as such. This should be changed to alpha + beta*xpred (generated quantities block). Un-altered, ypred is calculated for whole 'training' data (for each sampled alpha, beta), and eventhough this could be used for other purposes (residuals etc.) the purpose is to get a proper predictive distribution - that is, for a single year (2019).

I will give the fixed code in whole in the 3. task.

2. Figuring out the scale (tau) for prior.

We can use standard normal to poll the necessary scale. I'm using r's qnorm to get the values for the standard case:

```
a <- qnorm(c(0.005,0.995), mean = 0, sd = 1)
a
```

```
## [1] -2.575829  2.575829
```

We can now divide the given quantile information to get the desired deviation:

```
tau <- 69/a[2]
tau
```

```
## [1] 26.78749
```

And just for sanity-check, let's test everything looks as it should:

```
qnorm(c(0.005,0.995), mean = 0, sd = tau)
```

```
## [1] -69  69
```

Everything seems to be in order. And so τ=26.7874893. That being said, this seems somewhat loose constraint on the model (indeed, weakly informative).

3. Fixed Stan implementation.

Finally, here's the proper stan code. I write it on a separate file from rstudio, and then run it separately.

```
write("// I'm writing STAN model from rstudio

      data {
        int<lower=0> N; // number of data points
        vector[N] x; // observation year
        vector[N] y; // observation number of drowned
        real xpred; // prediction year
        real tau;
      }
      parameters {
        real alpha;
        real beta;
        real<lower=0> sigma; //upper bound changed to lower bound
      }
      transformed parameters {
        vector[N] mu;
        mu = alpha + beta*x;
      }
      model {
        beta ~ normal(0, tau); //Here's the added prior
```

```
        y ~ normal(mu, sigma);
      }
      generated quantities {
        real ypred;
        ypred = normal_rng(alpha+beta*xpred, sigma); //changed xpred
      }

      ",

      "stan_model7_1.stan"

      )

#compiles the code
stanc("stan_model7_1.stan")
```

So, I added the prior for beta in model block. Also, real tau in data block (this could have been hard coded as well) and parameter sigma has now a lower bound of zero. Predictions are now done for xpred (2019) as they should (generated quantities block).

I'll run the model to check if everything works.

Here's the drowning data (plus year 2019 for prediction, and the earlier tau for beta's prior):

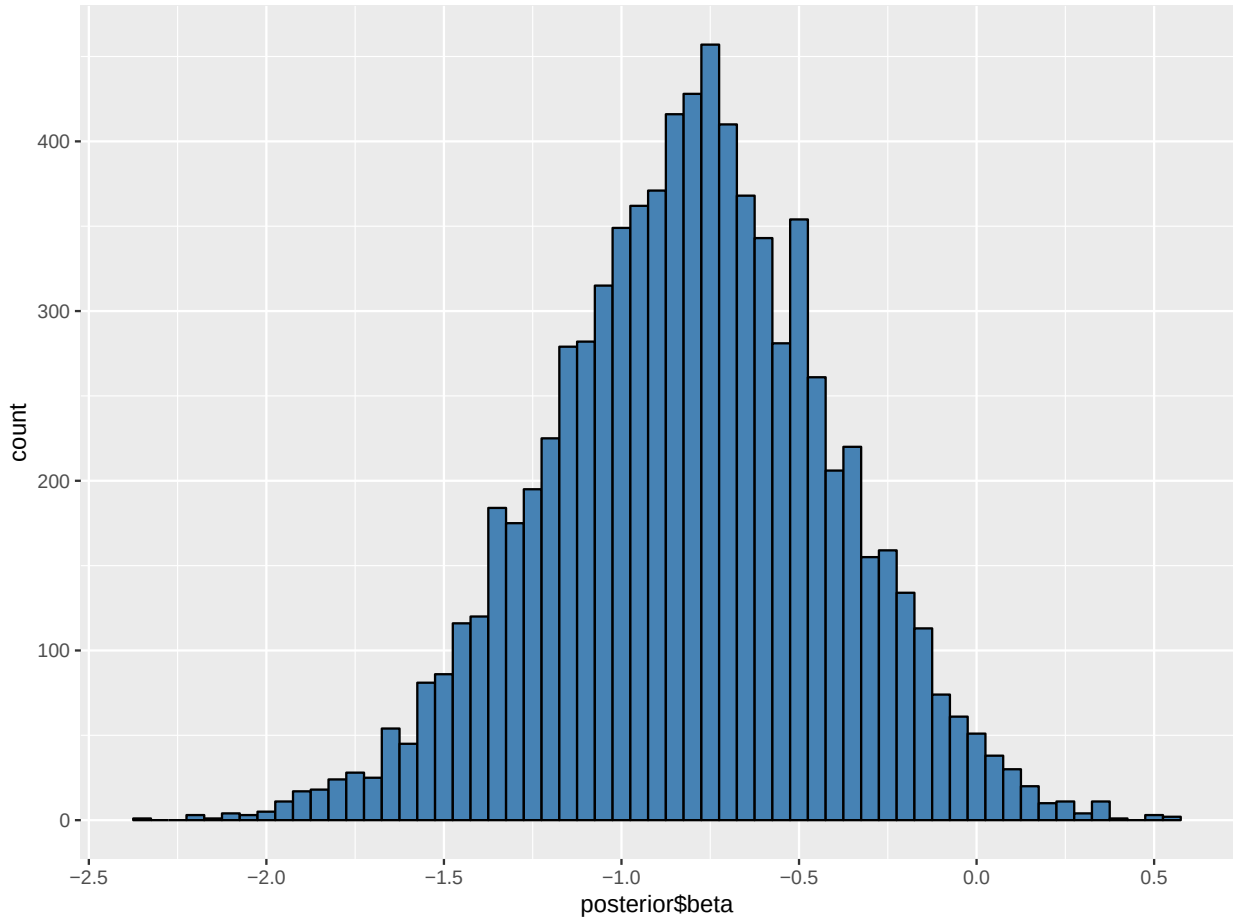
```
stan_data <- list(N=length(drowning$year), x=drowning$year, y=drowning$drownings,
                  xpred = 2019, tau = tau)

stan_model <- "./stan_model7_1.stan"
```

And here's a simple run:

```
fit <- stan(file = stan_model, data = stan_data, warmup = 500, iter = 2500, init = 'random')

# I'll extract the posterior for Rhat and plotting
posterior <- extract(fit)
```

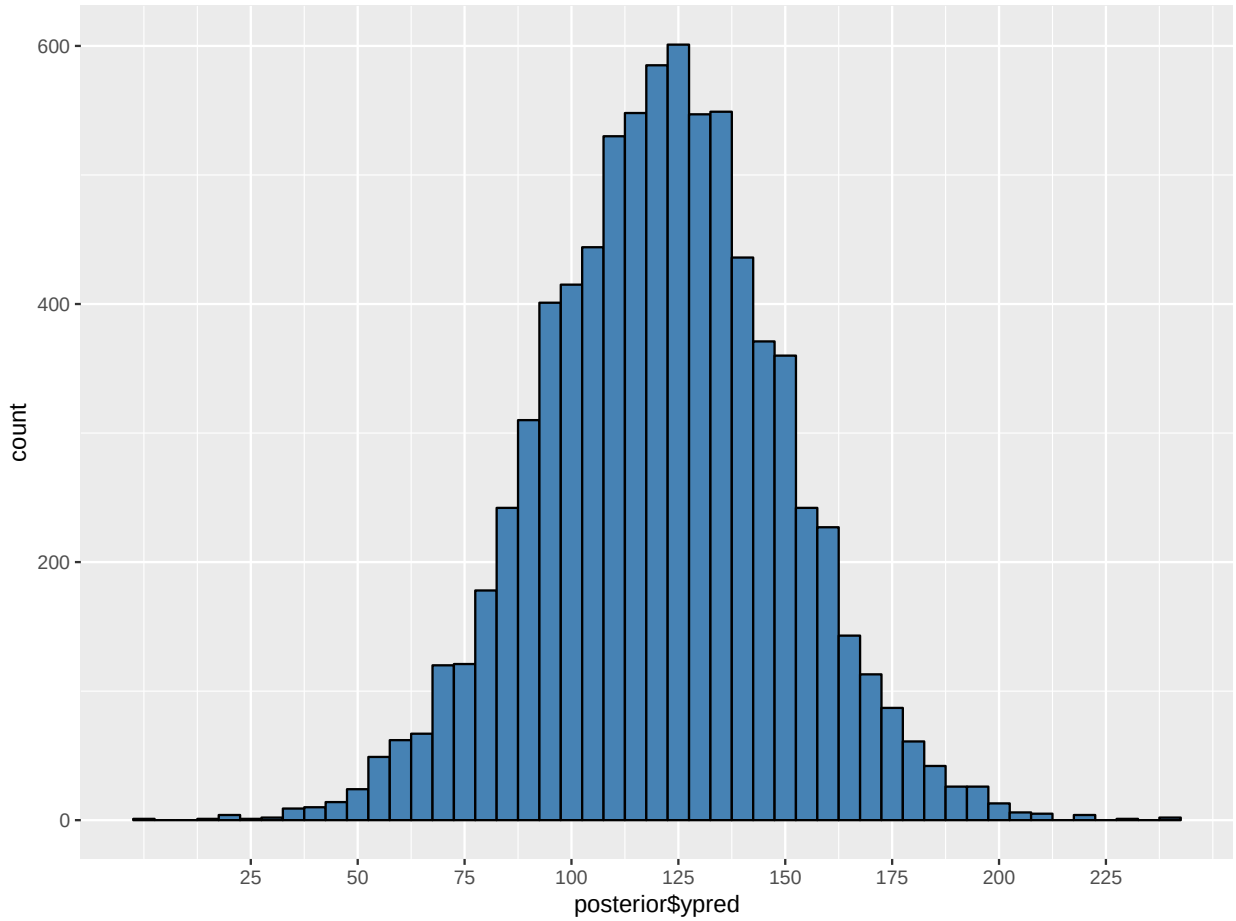
And here are the **Bayesian results**. Histogram for beta:

```
ggplot() +
  geom_histogram(aes(x=posterior$beta),binwidth = 0.05, fill = 'steelblue', color='black') +
  scale_x_continuous(breaks = c(-2.5,-2,-1.5,-1,-.5,0,.5,1))
```



Histogram for predictions:

```
ggplot() +
  geom_histogram(aes(x=posterior$ypred),binwidth = 5, fill = 'steelblue', color='black') +
  scale_x_continuous(breaks = seq(25,225,by = 25))
```



And expected values:

```
mean(posterior$beta)
```

```
## [1] -0.8109086
```

```
mean(posterior$ypred)
```

```
## [1] 120.9438
```

For comparison, a quick linear fit for the data (I only check beta value and prediction):

```
reg.model <- lm(drowning$drownings ~ drowning$year)
reg.model$coefficients[2]
```

```
## drowning$year
##    -0.8176861
```

```
(reg.model$coefficients[1]+ reg.model$coefficients[2]*2019) %>% as.vector()
```

```
## [1] 120.8556
```

As expected with such a loose prior, the results hardly differ. In any case, it seems the number of drowning deaths is decreasing :-)

BDA - Assignment 7 (part 2)

Anonymous

2 11 2019

Contents

Used libraries:

```
library(dplyr)
library(ggplot2)
library(rstan)
library(gdata)
library(bayesplot)
library(aaltobda)
data("factory")
```

```
options(mc.cores = 1)#parallel::detectCores())
rstan_options(auto_write = TRUE)
```

2. Hierarchical model, factory data with Stan

I was pretty lost with this one if I'm honest. The models them selves were not hard. But I'm still somewhat unclear what inferences I was supposed to do... Still, here's my best effort. I'm happy to take any pointers concerning any part of the task. Thank you.

I will write all models, pooled/separate/hierarchical, in a single stan file (it's faster to run and fiddle with it this way). And since they all use the same data, I will run them with a single swipe, and even though I don't need them all, I will take all predictions for each model. Then I will just analyze the ones that were required of us.

NOTE ABOUT PRIORS AND HYPERPRIORS: Even though we were given the permission to use uniform priors, I did use weakly informative ones for separate and hierarchical case. I hope the grader acknowledges this if the histograms seem different than the reference ones. Aki sent everyone an email explaining this issue. Also, the hierarchical model needed at least weakly informative priors for the simulation to behave well (without massive number of iterations).

CONCERNING THE STAN CODE: Idea for the models is as follows:

Pooled: Nothing special. I take all the data as a single group, single mu_P and sigma_P both from uniform. With posterior samples mu_P, sigma_P I then predict with normal_rng in generated quantities block. For pooled model, there is no difference between groups, so in that sense no prediction for a specific group (6). But the prediction is for mean of quality measurements in a general.

Separate: Now each group has it's own model (each mean quality measure has its own distribution) and own mu_S, sigma_S. Mu is drawn from uniform and sigma from scaled inverse chi^2. The scale for chi is roughly 2 times the sample sd (from data). Posterior samples are drawn with normal_rng in the generated quantities block *for each group*. I will use both the 6th group and all groups together for prediction for 6th and 7th (correspondingly).

Hierarchical: Groups have their own mu_H which are pulled from N(mu_0_H, sigma_0_H) with common hyperpriors mu_0_H ~ N(90,50) and sigma_0_H ~ scaled_inv_chi^2(K-1, 8) (K is group size, 8 comes

from data and experiment ~ twice the sample sd). In addition, all groups share sigma_H which has weakly informative cauchy(0,8) prior. Posterior predictions are drawn in a similar fashion as for separate model.

The models(P,S,H) don't share any parameters, only data (in stan code). Input array x has identifiers for different groups (pooled doesn't need it) and y has all quality measure data in order; 1,2,3,4,5,6,1,2,3,...

And the model(s): (everything is commented)

```
write("// I'm writing STAN model from rstudio

      data {
      //COMMON DATA

        int<lower=0> N;
        int<lower=0> K;
        int<lower=0, upper=K> x[N];
        vector[N] y;
      }
      parameters {
        //PARAMS FOR POOLED
        real mu_P;
        real<lower=0> sigma_P;

        //PARAMS FOR SEPARATE
        vector[K] mu_S;
        vector<lower=0>[K] sigma_S;

        //PARAMS FOR HIERACHICAL
        real mu_0_H;
        real<lower=0> sigma_0_H;
        vector[K] mu_H;
        real<lower=0> sigma_H;
      }
      transformed parameters {

      }
      model {
      //MODEL FOR POOLED (P)
        y ~ normal(mu_P, sigma_P); //uniform priors

      //MODEL FOR SEPARATE (S)
        sigma_S ~ scaled_inv_chi_square(K-1,8); //weak prior
        y ~ normal(mu_S[x], sigma_S[x]);

      //MODEL FOR HIERARCHICAL (H)
        mu_0_H ~ normal(90,50); // weak hyperprior
        sigma_0_H ~ scaled_inv_chi_square(K-1,8); // weak hyperprior
        sigma_H ~ cauchy(0,8); // weak prior
        mu_H ~ normal(mu_0_H,sigma_0_H); // prior
        y ~ normal(mu_H[x], sigma_H);
      }
      generated quantities {
        real ypred_H[6];
        real ypred_S[6];
        real ypred_P;
```

```
        //PREDICTION FOR POOLED
        ypred_P = normal_rng(mu_P, sigma_P);

        //PREDICTION FOR SEPARATE
        ypred_S = normal_rng(mu_S, sigma_S);

        //PREDICTION FOR HIERACHICAL
        ypred_H = normal_rng(mu_H,sigma_H);
      }

      ",

      "stan_model7_all.stan"

      )

#compiles the code
stanc("stan_model7_all.stan")
```

Then the data. Scale_sigma shows how I reasoned the scale for cauchy and inv chi priors. Experimenting agreed.

```
scale_sigma <- sqrt(var(c(t(factory[,1:6]))))/5)


all_data <-list(N = 6*nrow(factory),
              K = 6,
              x = rep(1:6, nrow(factory)),
              y = c(t(factory[,1:6])))


stan_model_all <- "./stan_model7_all.stan"
```

And the run. Four chains did suffice with weak priors. Warmup period is on the longer side, but thats mostly an added insurance. Overall 8000 draws are produced for each posterior mu and prediction.

```
fit <- stan(file = stan_model_all, data = all_data, warmup = 1000, iter = 3000,
                      control=list(adapt_delta=0.98))
```

Extracting the fit and monitoring the results (mostly Rhat is of interest):

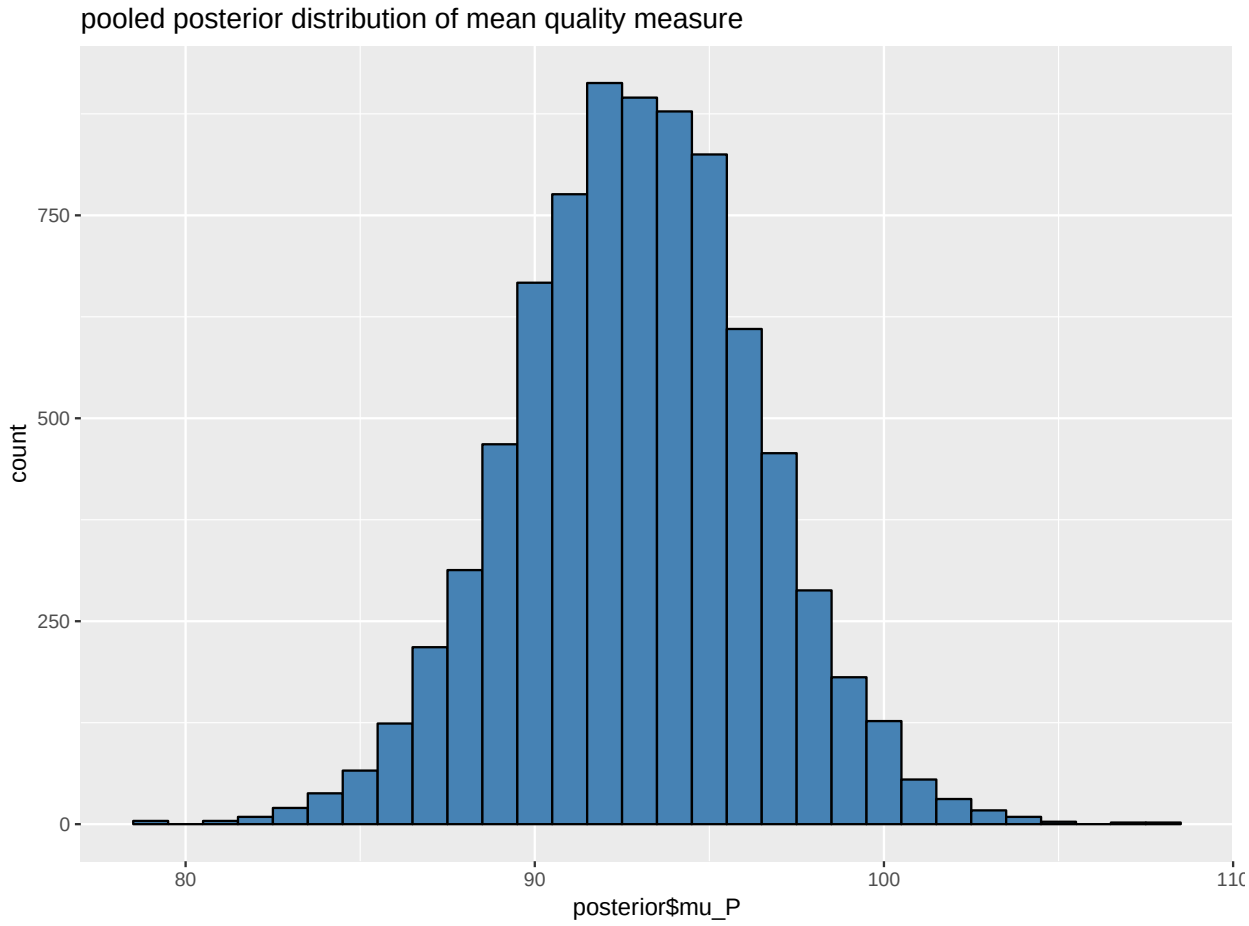
```
posterior <- extract(fit)
m <- monitor(fit)
```

Then the separate histograms.

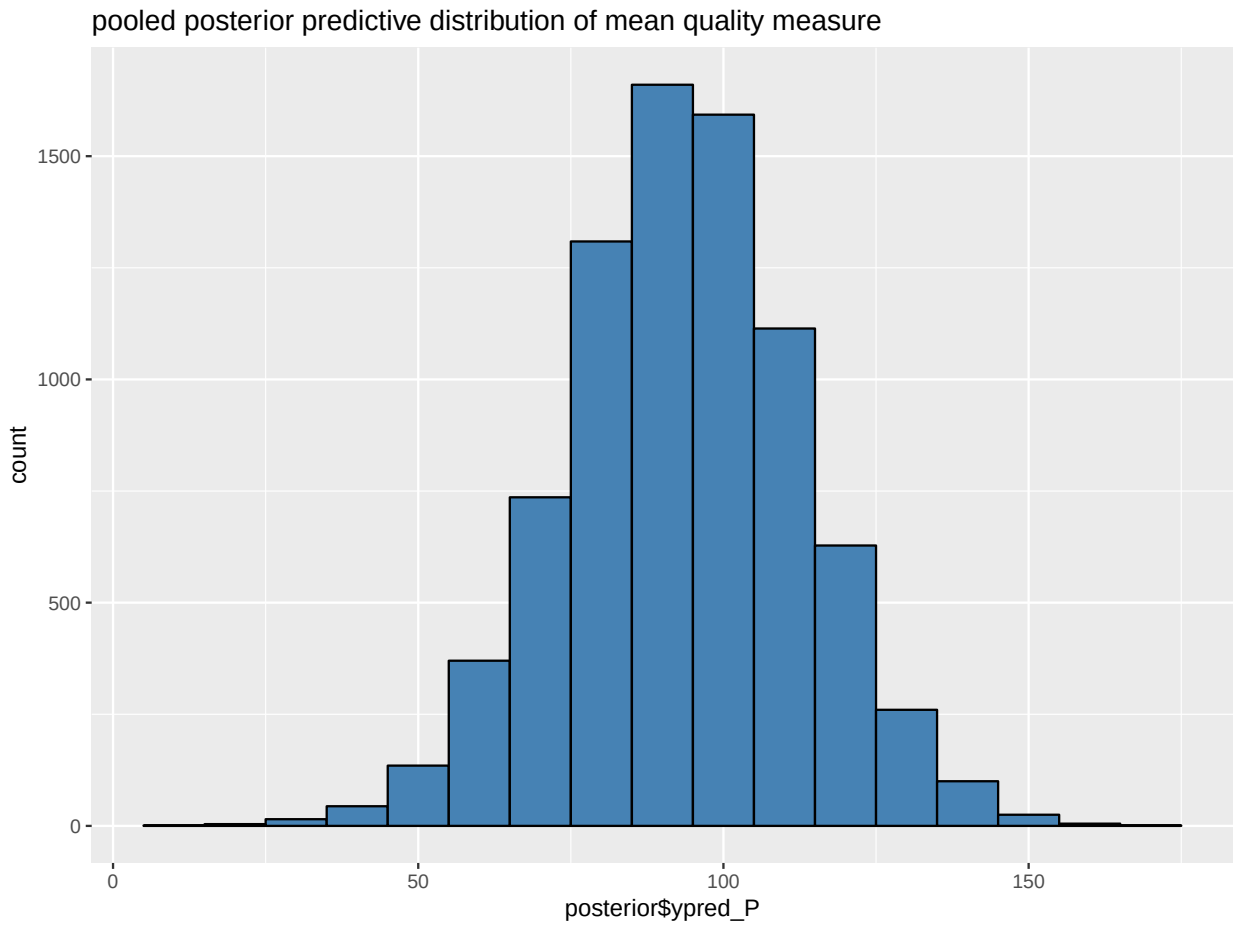
Histogram for pooled. There is only a posterior distribution for the whole group, no differentiation between groups. So the only reasonable histograms from these data, are posterior distribution of the mean of quality measurement (MQM from now on) and posterior predictive. Now, the posterior predictive is of course what can be considered as the prediction of the seventh machines MQM. But for a specific machine (6) there is no separate posterior. However, had I no other information, the posterior distribution would still be the best *available* option for posterior for the 6th. So, I will plot both posterior mu_P and posterior predictive ypred_P.

So, in some sense the pooled posterior distributions are for “a” machine, un-specified.

```
ggplot() +
  geom_histogram(aes(x=posterior$mu_P),binwidth = 1, fill = 'steelblue', color='black') +
  ggtitle('pooled posterior distribution of mean quality measure' )
```




```
  #scale_x_continuous(breaks = c(-2.5,-2,-1.5,-1,-.5,0,.5,1))

ggplot() +
  geom_histogram(aes(x=posterior$ypred_P),binwidth = 10, fill = 'steelblue', color='black') +
  ggtitle('pooled posterior predictive distribution of mean quality measure' )
```

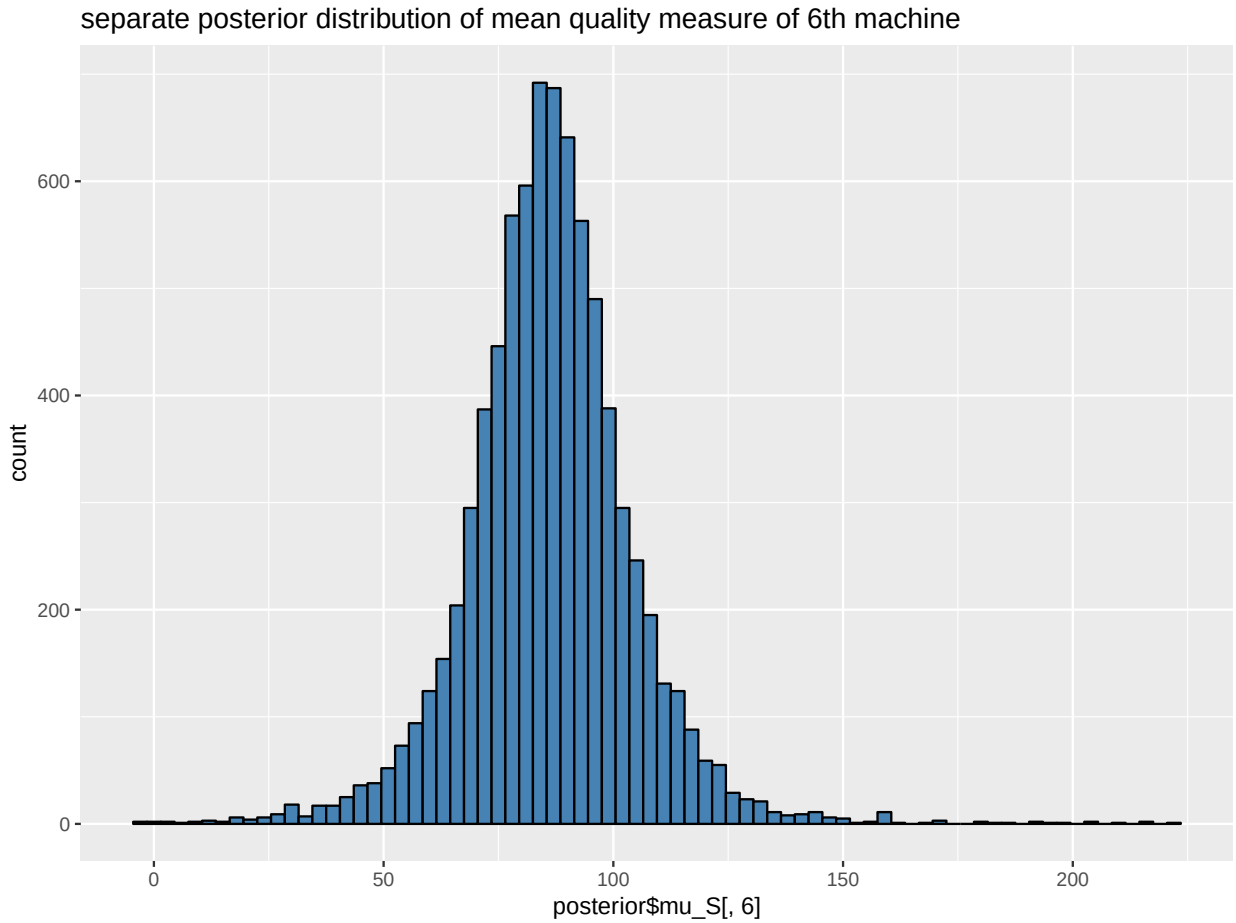



```
  #scale_x_continuous(breaks = c(-2.5,-2,-1.5,-1,-.5,0,.5,1))
```

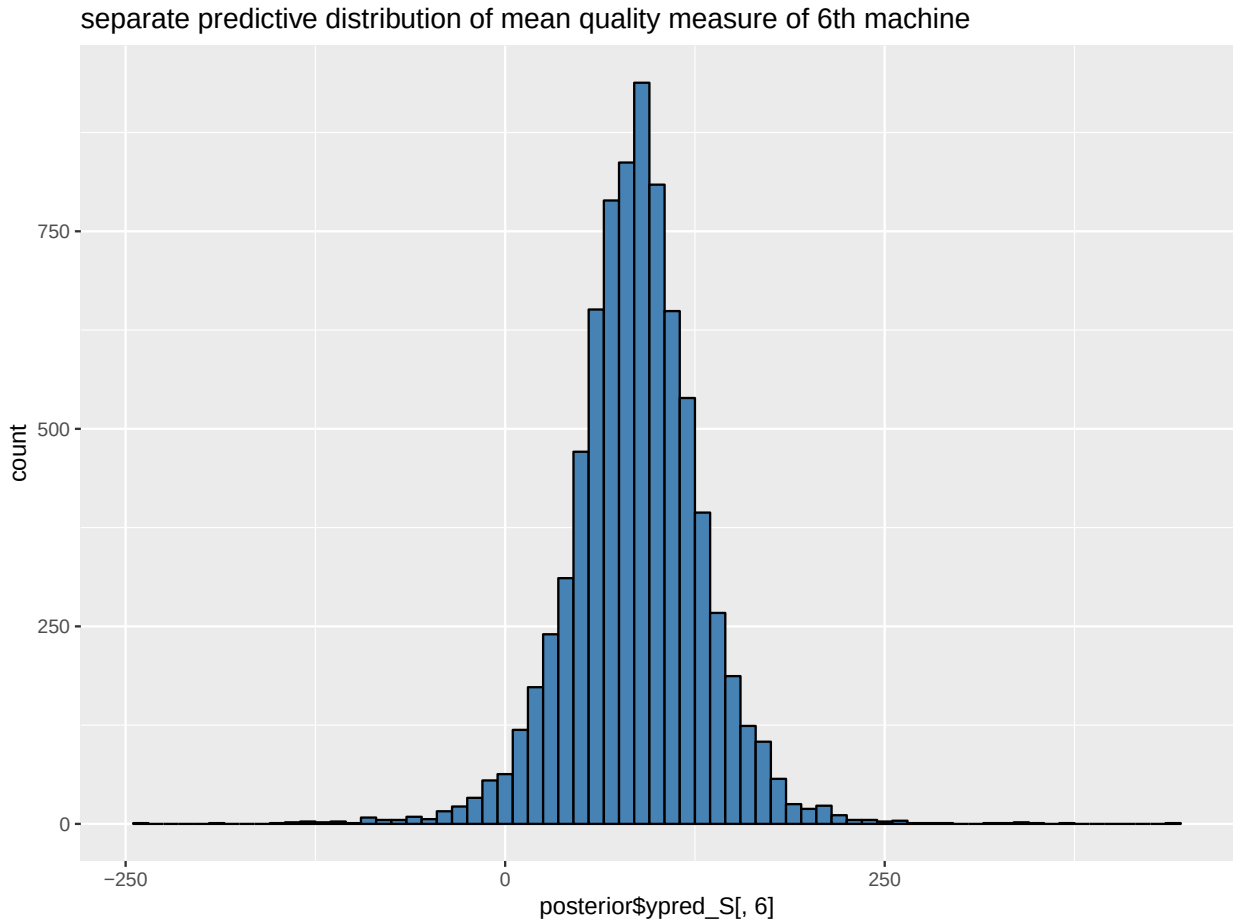
The posterior MQM is quite narrow (roughly between 85-100) for pooled model, and the posterior predictive is a bit wider (roughly between 50-130) since the pooled groups had some variance in between their means. None of these are for a specific machine, but for an unknown/un-specified machine. So in some sense they are both for the 6th and unknown/future machines (and all others).

Histograms for separate. For separate case there are histograms for all that was asked. For the seventh machine (prediction) I will take sample over all posterior samples (I narrow the xlims abit, to give better comparison for others - some individual tail samples get dropped - and rstudio complains a little. Not particularly important here).

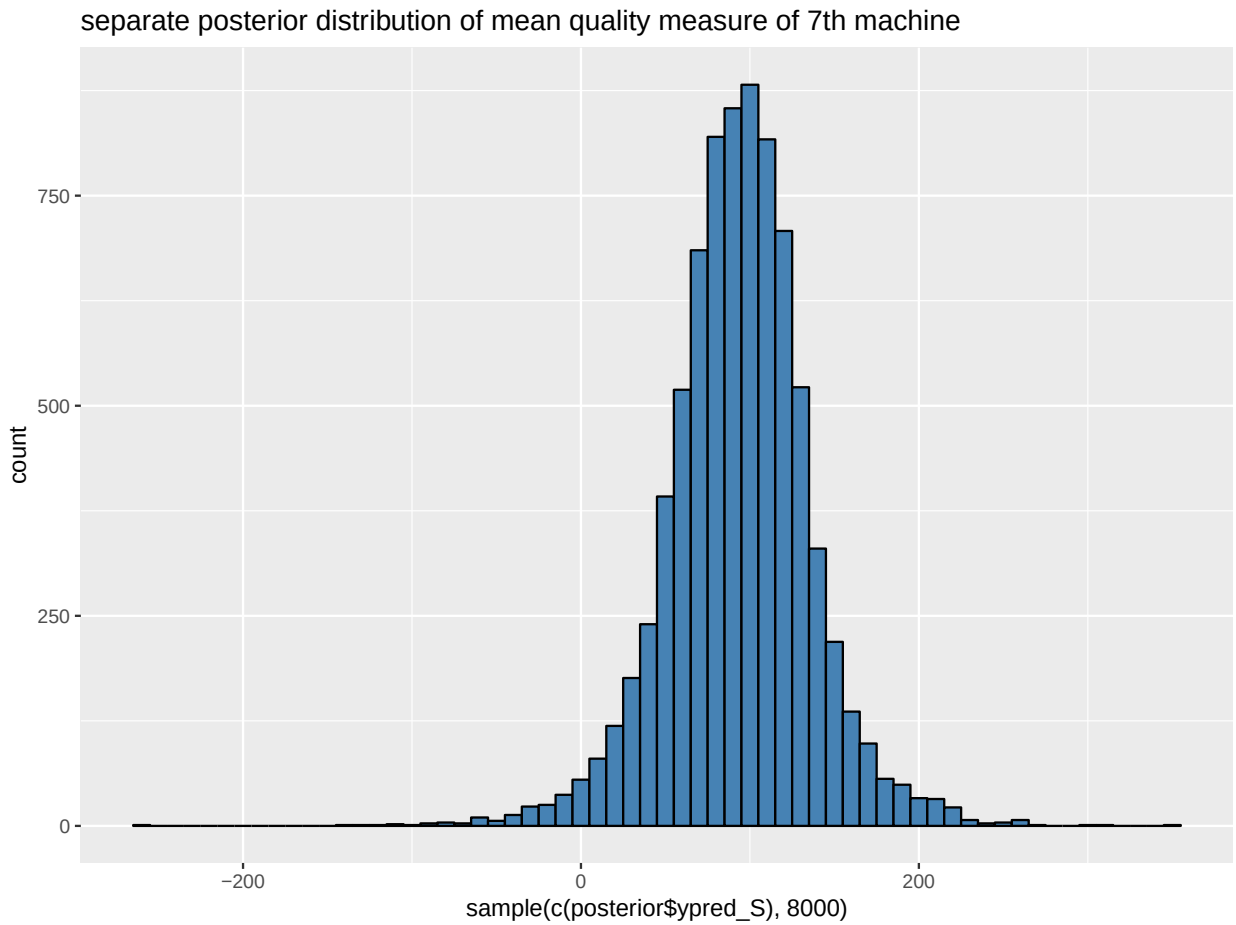
```
#POSTERIOR FOR 6th
ggplot() +
  geom_histogram(aes(x=posterior$mu_S[,6]),binwidth = 3, fill = 'steelblue', color='black') +
  ggtitle('separate posterior distribution of mean quality measure of 6th machine' )
```




```
  #scale_x_continuous(breaks = c(-2.5,-2,-1.5,-1,-.5,0,.5,1))

#PREDICTIVE FOR 6th
ggplot() +
  geom_histogram(aes(x=posterior$ypred_S[,6]),binwidth = 10, fill = 'steelblue', color='black') +
  ggtitle('separate predictive distribution of mean quality measure of 6th machine' )
```




```
  #scale_x_continuous(breaks = c(-2.5,-2,-1.5,-1,-.5,0,.5,1))

#POSTERIOR OVER ALL (RANDOM SAMPLE OF SIZE 8000)
ggplot() +
  geom_histogram(aes(x=sample(c(posterior$ypred_S),8000)),binwidth = 10, fill = 'steelblue', color='bla
  ggtitle('separate posterior distribution of mean quality measure of 7th machine' )# +
```

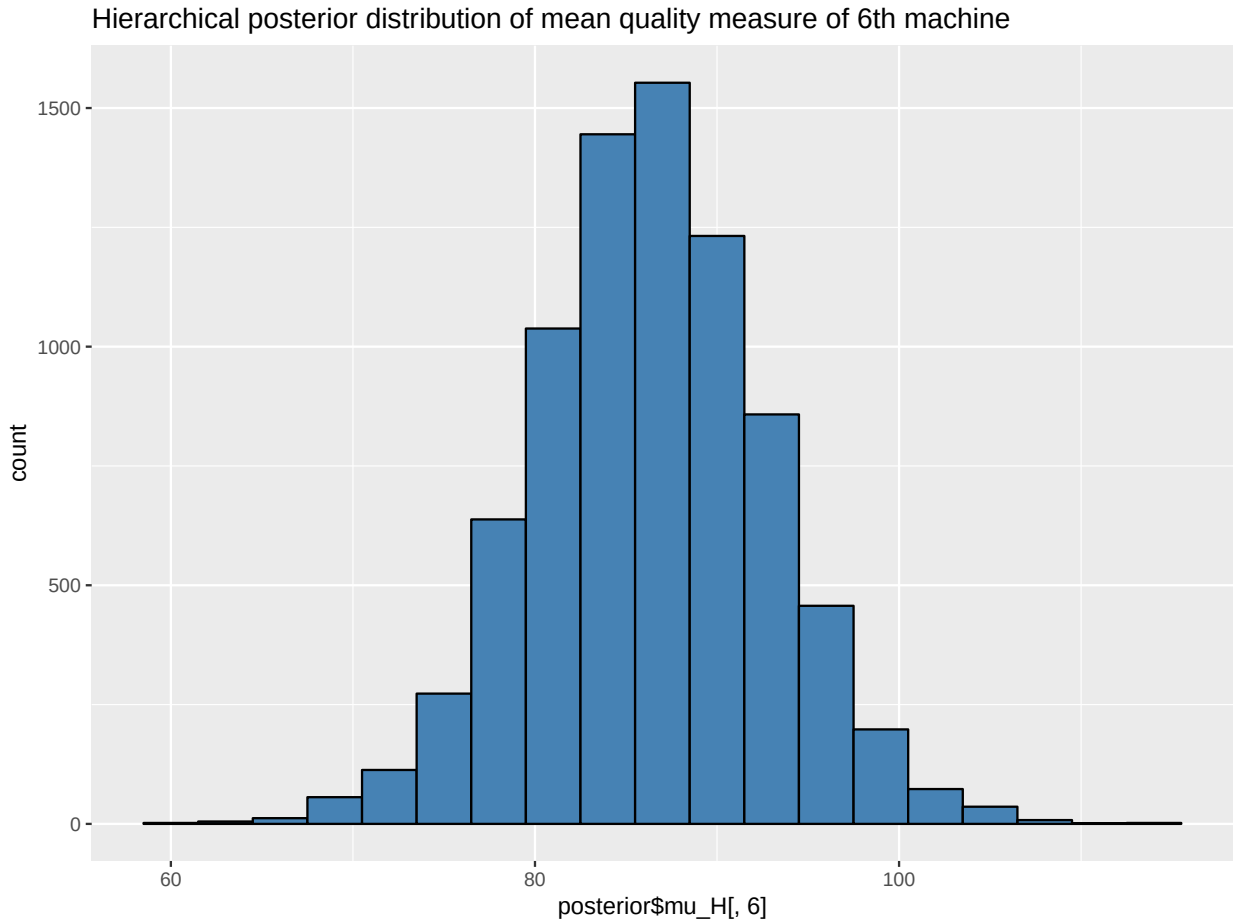



```
  #xlim(c(0,200))
  #scale_x_continuous(breaks = c(-2.5,-2,-1.5,-1,-.5,0,.5,1))
```

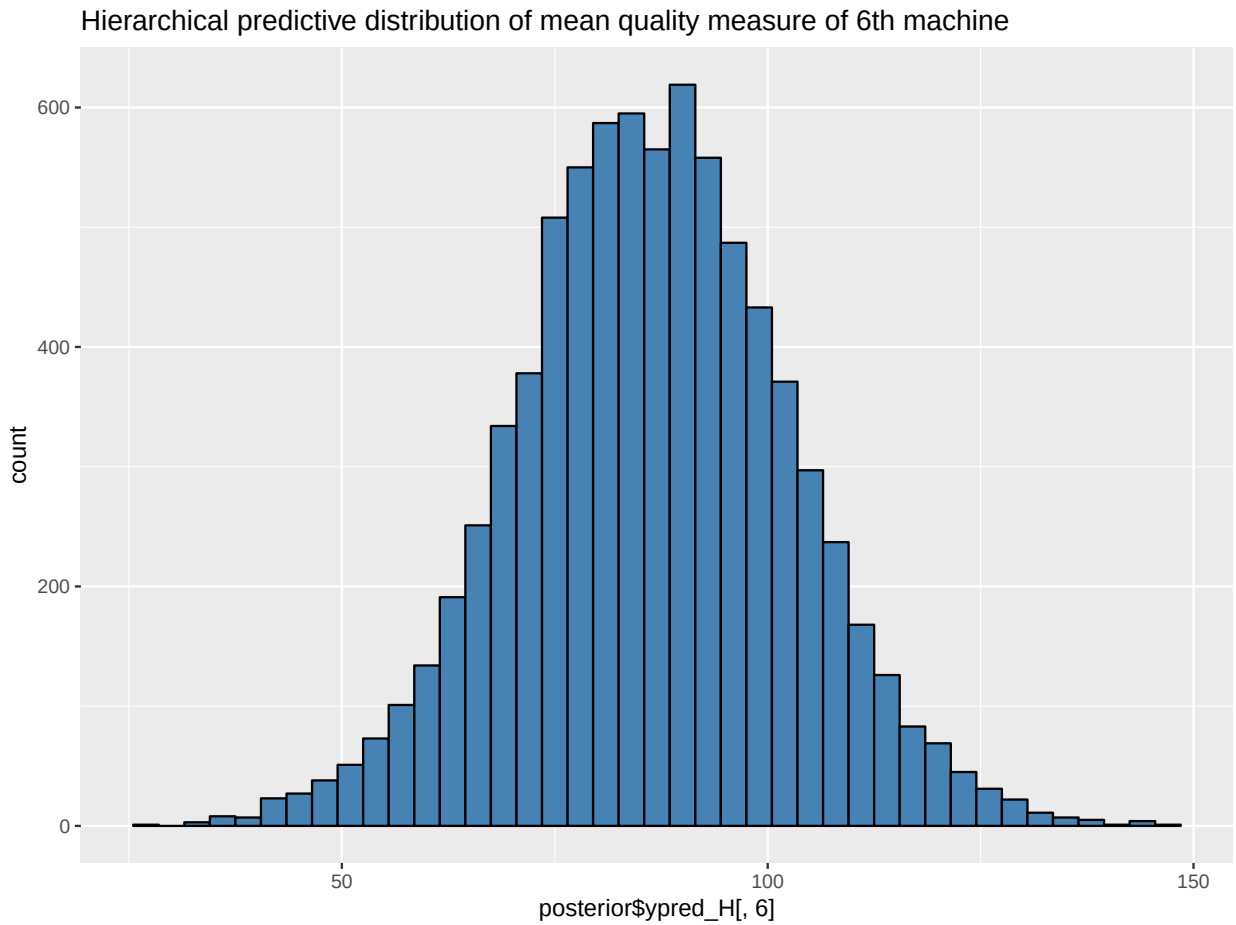
Separate posterior for sixth machine is pretty wide (roughly between 40-130), and the predictive for 6th is even wider. And due to varying group means, so is the posterior over all of them (0-200).

Histograms for hierarchical. Same as with separate, with hierarchical model we can also produce all the required histograms.

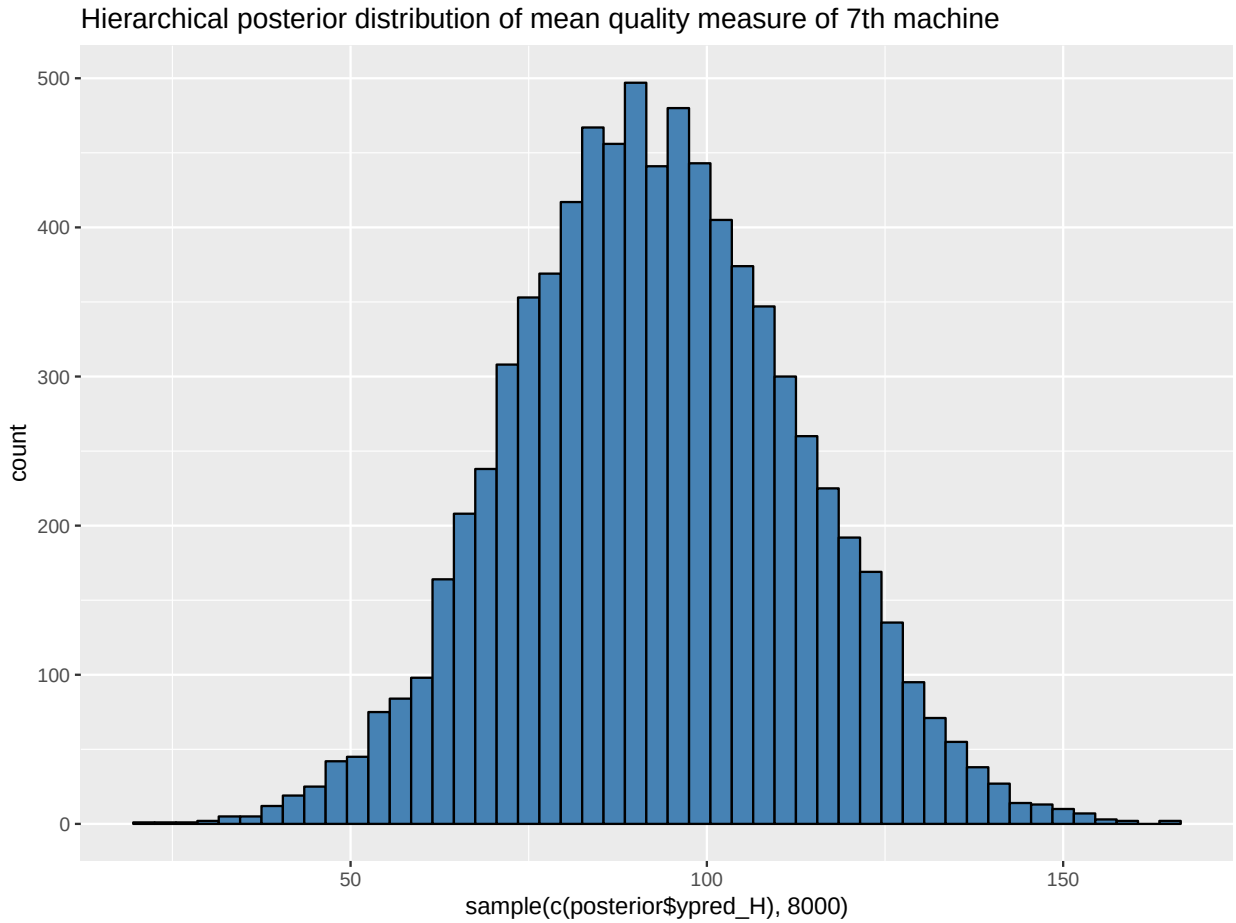
```
ggplot() +
  geom_histogram(aes(x=posterior$mu_H[,6]),binwidth = 3, fill = 'steelblue', color='black') +
  ggtitle('Hierarchical posterior distribution of mean quality measure of 6th machine' )
```




```
  #scale_x_continuous(breaks = c(-2.5,-2,-1.5,-1,-.5,0,.5,1))


ggplot() +
  geom_histogram(aes(x=posterior$ypred_H[,6]),binwidth = 3, fill = 'steelblue', color='black') +
  ggtitle('Hierarchical predictive distribution of mean quality measure of 6th machine' )
```




```
  #scale_x_continuous(breaks = c(-2.5,-2,-1.5,-1,-.5,0,.5,1))



ggplot() +
  geom_histogram(aes(x=sample(c(posterior$ypred_H),8000)),binwidth = 3, fill = 'steelblue', color='blac
  ggtitle('Hierarchical posterior distribution of mean quality measure of 7th machine' )
```




```
  #scale_x_continuous(breaks = c(-2.5,-2,-1.5,-1,-.5,0,.5,1))
```

Hierarchical posterior distribution for 6th machine (roughly between 70-110) is still wider than in the pooled model, but significantly narrower than with separate model. Same applies for posterior predictive of the 6th machine (roughly between 60-120). And in the similar way, the posterior for seventh (unknown) machine (roughly between 50-130)is still narrower than separate but wider than the pooled one.

It is somewhat hard to compare all of these visually. Here's the result from monitoring. It gives a good idea how the different models look like (in particular the means, sd's and quantile information shows how these models compare). It also shows that all the chains have converged, and the sampling is likely to be valid. All Rhat values are close to one.

In summary: With pooled model, we get relatively narrow distributions (both for posterior and prediction). For separate models, since there is no information flow between the models, we get more dispersed results. For the separate 6th machine this spread is due to smaller set of data, and this is only in part compensated by group/machine particular statistic. And over all separate models, even though there is more data, the data is more dispersed (since the models were fully separated, i.e., no information flow). For the hierarchical model, even though the groups are still separate and varied, the models can change information via hyperparams (and therefore priors). This way the whole data set can be utilized whilst differentiating between machines. The data of one machine only influences the posterior of another machine through the priors.

```
m %>% as.data.frame() %>% select(mean, sd, `2.5%`, `97.5%`)
```

```
##                mean        sd        2.5%      97.5%
## mu_P       92.91238  3.489408   86.040573   99.69361
## sigma_P    18.80496  2.609733   14.468502   24.54667
```

```
## mu_S[1]      75.74249 18.014461   39.054130  111.73043
## mu_S[2]     106.44821 16.066247   73.694789  137.96525
## mu_S[3]      87.90740 17.091361   55.225911  122.80098
## mu_S[4]     111.55357 14.341391   82.366190  141.15021
## mu_S[5]      89.56044 15.538257   57.250927  119.87902
## mu_S[6]      86.20106 18.127546   49.759765  122.15712
## sigma_S[1]   37.50822 15.782505   18.505092   77.71223
## sigma_S[2]   32.59622 15.376908   15.016468   70.20695
## sigma_S[3]   33.70231 15.852113   15.612941   74.60868
## sigma_S[4]   30.49933 13.694536   13.928378   65.58122
## sigma_S[5]   31.77778 14.239391   15.216996   66.27778
## sigma_S[6]   37.08498 15.777676   18.297157   76.76012
## mu_0_H       92.81676 12.282287   67.404336  117.44956
## sigma_0_H    29.66422 11.526065   15.065088   56.93181
## mu_H[1]      77.15699  6.521751   64.475935   90.21681
## mu_H[2]     105.31072  6.578320   92.147699  118.18215
## mu_H[3]      88.20230  6.323778   75.575992  100.87390
## mu_H[4]     110.43775  6.581920   97.784469  123.36595
## mu_H[5]      90.21318  6.318321   77.732328  102.59536
## mu_H[6]      86.31621  6.385464   73.713775   98.86922
## sigma_H      14.54729  2.114159   11.039943   19.37334
## ypred_H[1]   77.23281 16.190065   45.936233  108.70211
## ypred_H[2]  105.37152 16.192131   73.917856  137.59800
## ypred_H[3]   87.99619 16.075323   56.782011  119.06477
## ypred_H[4]  110.26801 16.328817   77.376834  142.03641
## ypred_H[5]   90.12501 16.193980   58.245024  122.04282
## ypred_H[6]   86.33215 16.015648   54.352018  118.36824
## ypred_S[1]   75.39837 45.580474  -13.769288  169.06657
## ypred_S[2]  106.39089 39.182120   29.603660  186.78326
## ypred_S[3]   87.27466 40.228038    6.583603  167.56774
## ypred_S[4]  111.62705 36.158778   38.951102  186.37230
## ypred_S[5]   89.72674 37.908085   14.713820  164.72265
## ypred_S[6]   86.46360 43.268839    1.145539  171.33829
## ypred_P      93.34093 19.256769   55.064557  131.39099
## lp__       -422.97511  3.833697 -431.556243 -416.62617
```

You can see the means for our statistics from monitor data:

- mu_P ~ 93 with sd 3.5
- mu_S[6] ~ 86 with sd 18.7 (more spread out than pooled one)
- mu_H[6] ~ 86 with sd 6.2 (in between)

Also for predictions for 6th:

- ypred_P ~ mean 93 with sd 19.3 (note: this is not predicting for a specific machine)
- ypred_S[6] ~ mean 86 with sd 44.5
- ypred_H[6] ~ mean 86 with sd 16

Also:

```
#Hierarchical posterior over all groups
mean(c(posterior$ypred_H))
```

```
## [1] 92.88761
```

```
sd(c(posterior$ypred_H))
```

```
## [1] 19.77153
```

All and all these did behave as expected. I found that with different prior choices there was some variation in the results, however, the nature of the results (and how the models compared) did not change significantly.