title: "BDA - Assignment 7"
author: "Anonymous"
output:
pdf_document:
toc: yes
toc_depth: 1

```
# To install aaltobda, see the General information in the assignment.
library(aaltobda)
library("ggplot2")
library(rstan)
```

```
## Loading required package: StanHeaders

## rstan (Version 2.21.2, GitRev: 2e1f913d3ca3)

## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

1. Linear model: drowning data with Stan (3p)

The provided data drowning in the aaltobda package contains the number of people who died from drowning each year in Finland 1980–2019. A statistician is going to fit a linear model with Gaussian residual model to these data using time as the predictor and number of drownings as the target variable (see the related linear model example for the Kilpisj¨arvi-temperature data in the example Stan codes). She has two objective questions:

i) What is the trend of the number of people drowning per year? We would plot the histogram of the slope of the linear model.
ii) What is the prediction for the year 2020? We would plot the histogram of the posterior predictive distribution for the number of people drowning at ~x = 2020.

Your task is to fix the Stan code to be able to run this linear regression model.

The data for the assignment.

```
data("drowning")
```

1.

The three mistakes in the code and how I fixed them:

1. The first mistake was that there was no semi-colon at the end of the row where the model was defined. Fix: Added the the semi-colon.
2. The standard deviatoin parameter sigma had upper bound set. There should not be a upper bound for this parameter. Only contraint is that it cannot be negative. Fix: Change upper bound constraint to lower bound constraint.
3. The quantity ypred is a real number drawn from a normal distribution. There was mu as a parameter for the normal distribution in the initial code. This caused error, because mu is a vector of dimension N. Fix: For ypred, we need to use xpred when transforming parameter mupred. This way, we get real

number mupred, which we can give as a parameter to normal distribution where the ypred is to be sampled from.

The fixed mistakes are also commented on the corresponding lines within the Stan model code.

2.

Determining suitable value for σ_β. The approximate numerical value for σ_β is

```
bound <- 69
sigma_quantile <- qnorm(c(0.005, 0.995))
sb <- bound / sigma_quantile
sb[2]
```

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

3.

Adding the desired beta prior in the Stan code:

Added line …

real sigma_beta ;

in parameters block.

Added line …

beta ~ normal (0, sigma_beta) ; // Weakly informative prior for beta added here.

… in model block.

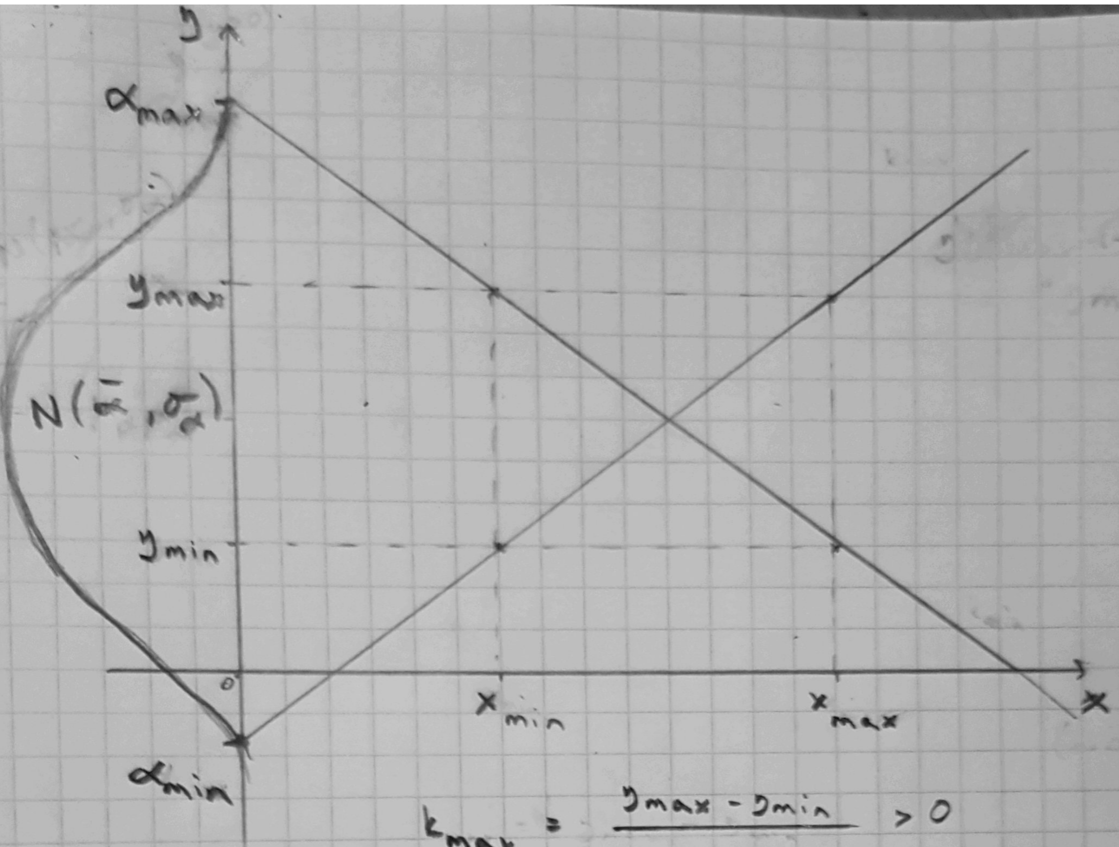
4.

Adding the weakly informative prior for the intercept alpha.

If we approach the problem of determining weakly informative alpha (normal) prior in a similar manner as was done in the part 2 for beta prior, we need to figure out the crude estimates, between which two values the intercept lies with high probability. We can use, for example the same probability of 0.99 that used in part 2.

I added a hand-made plot which visualizes the bounds for intercept alpha.

```
knitr::include_graphics("BDA_7.jpg")
```



$$k_{max} = \frac{y_{max} - y_{min}}{x_{max} - x_{min}} > 0$$

$$k_{min} = \frac{y_{min} - y_{max}}{x_{max} - x_{min}} < 0$$

$$\alpha_{max} = k_{min} \cdot (-x_{min}) + y_{max}$$

$$\alpha_{min} = k_{max} \cdot (-x_{min}) + y_{min}$$

$$\rightarrow \alpha = N(\bar{\alpha}, \sigma_\alpha)$$

So basically, the boundaries for the alpha intercept can be approximated by setting two lines that have the max and min slope coefficients based on the data. With 0.99 probabilty, we can say that the intercept for our model is something betweeen the boundaries. Based on this weak information about the intercept boundaries, we can compute the required parameters for our normal weakly informative prior form where alpha can be sampled.

Now we need to implement the calculations in R, based on the plot/sketch above. The approximate numerical value for μ_α is

```
# Calculate the min and max values for both x and y
y_min <- min(drowning$drownings)
y_max <- max(drowning$drownings)
x_min <- min(drowning$year)
x_max <- max(drowning$year)

# Decide the maximum and minimum slope coefficient
k_max <- (y_max - y_min) / (x_max - x_min)
k_min <- (y_min - y_max) / (x_max - x_min)

# Compute maximum and minimum values for intercept bound where we assume the intercept to lie with 0.99
alpha_min = k_max * (-x_min) + y_min
alpha_max = k_min * (-x_min) + y_max

# Prior of alpha is normal distribution, so we need to compute mu_alpha.
# It is the mean of the boundary values.
ma <- mean(c(alpha_min, alpha_max))
ma
```

```
## [1] 143
```

The approximate numerical value for σ_α is

```
# Bound is decided with taking the difference between the mean of the normal distribution and the minim
bound <- ma - alpha_min
# Now we calculate the sigma_alpha similarly as we calculated sigma_beta in part b.
sigma_quantile <- qnorm(c(0.005, 0.995))
sa <- bound / sigma_quantile
sa[2]
```

```
## [1] 2456.296
```

Adding the desired alpha prior in the Stan code:

Added lines …

real sigma_alpha ; real mu_alpha ;

… in paramters block.

Added line …

alpha ~ normal (mu_alpha, sigma_alpha) ; // Weakly informative prior for alpha added here.

… in model block.

Add all the necessary data into one list.

```
drowning_data <- list(N = dim(drowning)[1], x = drowning$year, y = drowning$drownings, xpred = 2020, si
```

THE STAN MODEL:

```
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 sigma_alpha ;
  real mu_alpha ;
  real sigma_beta ;
}
parameters {
  real alpha ;
  real beta ;
  real <lower = 0> sigma ; // Mistake 2. The upper bound changed to lower bound.
}
transformed parameters {
  vector [N] mu = alpha + beta * x ;
  real mupred = alpha + beta *  xpred ; // Mistake 3. Set parameter mupred.
}
model {
  alpha ~ normal (mu_alpha, sigma_alpha) ; // Weakly informative prior for alpha added here.
  beta ~ normal (0, sigma_beta) ; // Weakly informative prior for beta added here.
  y ~ normal (mu , sigma) ; // Mistake 1. There was no semi-colon here.
}
generated quantities {
  real ypred = normal_rng (mupred, sigma); // Mistake 3. Use mupred instead of mu.
}
```

```r
fit <- sampling(drowning_model, drowning_data, seed = 123, refresh=0)
```

```
## Warning: There were 1098 transitions after warmup that exceeded the maximum treedepth. Increase max_
## http://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded

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

```r
monitor(fit, print = FALSE)[c(1, 2, 3, 42), c(1, 3, 10)]
```

```
##                 mean          sd     Rhat
## alpha 1621.8743344 770.7885203 1.005643
## beta    -0.7426369   0.3857503 1.005679
## sigma   26.1090592   3.3155270 1.006011
## ypred  121.5401351  27.2240572 1.000795
```

So, the answers for the two objective question are…

i) The trend of the number of people dying each year is negative, so each year less people are drowning.
ii) Prediction for the year 2020 is 122 if rounded upwards.

2. Hierarchical model: factory data with Stan (3p)

The factory data in the aaltobda package contains quality control measurements from 6 machines in a factory (units of the measurements are irrelevant here). In the data file, each column contains the measurements for a single machine. Quality control measurements are expensive and time-consuming, so only 5 measurements were done for each machine. In addition to the existing machines, we are interested in the quality of another machine (the seventh machine).

```r
library(aaltobda)
data("factory")
```

Separate model

As it says in the assignment description in the separate model, each machine has its own model.

In the Stan github page (https://github.com/stan-dev/stan/wiki/Prior-Choice-Recommendations) it says that a good general weakly informative prior is normal distribution with mean equal to data mean and standard deviation equal to deviation between the column means:

```r
mu_prior_mu <- mean(colMeans(factory))
mu_prior_sigma <- sd(colMeans(factory))

round(mu_prior_mu, 1)
```

```
## [1] 92.9
```

```r
round(mu_prior_sigma, 1)
```

```
## [1] 13.4
```

On the other hand, on the course book BDA3, on page 55 it says:

"We characterize a prior distribution as weakly informative if it is proper but is set up so that the information it does provide is intentionally weaker than whatever actual prior knowledge is available."

It is also written on the same page that:

"Rather than trying to model complete ignorance, we prefer in most problems to use weakly informative prior distributions that include a small amount of real-world information, enough to ensure that the posterior distribution makes sense."

Based on this, we could choose a weakly informative prior to be e.g. $\mu_j \sim N(100, 50)$. Here the data mean is close to the actual value, but the second parameter is set, on purpose, higher than we know based on data it is. This way we have enough information to make sure that our inferences are contrained to be reasonable. We started from strongly informative data prior and broadened it to account for uncertainty in our prior beliefs and in the applicability of any historically based prior on new data. (BDA3, p. 55-56)

With this prior the resulting posterior will most likely make sense, but the prior isn't too informative.

Weakly informative prior for σ_j parameter needs to be decided bit differently. On BDA3 p. 130 it says that for variance parameters we should consider the t family of distributions (actually, the half-t, since the scale parameter τ is constrained to be positive) as an alternative class that includes normal and Cauchy as edge cases. For our purposes, it is enough to recognize that the half-Cauchy can be a convenient weakly informative family; the distribution has a broad peak at zero and a single scale parameter, which we shall label A to indicate that it could be set to some large value. It also says in the book that we shall consider half-Cauchy models for variance parameters which are estimated from a small number of groups (so thatinferences are sensitive to the choice of weakly informative prior distribution). In our case the data has only 5 groups so based on this half-Cauchy could be reasonable choice.

Later on pages 131 and 132 of the BDA3 it is also shown via example, that by choosing scale parameter value of the half-Cauchy distribution correctly, a good posterior is achieved and the whole model will perform well. The scale parameter value should be chosen to be a bit higher than we expect for the standard deviation of the underlying data, so that the model will be constrained only weakly. Based on this we choose our weakly informative prior for σ_j to be half-Cauchy with scale parameter 40 because we expect our data to deviate approximately 25 from data mean and we on purpose choose higher value than that.

Separate model described with mathematical notation

$$y_{ij} \sim N(\mu_j, \sigma_j)$$

$$\mu_j \sim N(100, 50)$$

$$\sigma_j \sim Cauchy(0, 40) > 0$$

SEPARATE MODEL WITH STAN

```
data {
  int < lower =0 > N; // n of measurements
  int < lower =0 > J; // n of machines
  vector[J] y[N];
}
parameters {
  vector<lower = 0>[J] mu ;
  vector<lower = 0>[J] sigma ;
}
model {
  // weakly informative priors
  for ( j in 1: J ){
    mu [j] ~ normal (100, 50);
    sigma [j] ~ cauchy(0, 40);
  }
  // likelihood
  for ( j in 1: J )
    y[ ,j ] ~ normal (mu[j], sigma[j]);
}
generated quantities {
  vector[J] ypred ;
  // Compute predictive distribution for J machines
  for (j in 1:J)
    ypred[j] = normal_rng (mu[j], sigma[j]);
}
```

```
stan_data <- list(y = factory,N = nrow(factory),J = ncol(factory))
sm <- rstan::sampling(separate_model, data = stan_data)
```

```
##
## SAMPLING FOR MODEL '736af21cee6aa08cea279ca878408707' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 9e-06 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.09 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.053299 seconds (Warm-up)
## Chain 1:                0.033094 seconds (Sampling)
## Chain 1:                0.086393 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '736af21cee6aa08cea279ca878408707' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 6e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.06 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.060041 seconds (Warm-up)
## Chain 2:                0.041333 seconds (Sampling)
## Chain 2:                0.101374 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '736af21cee6aa08cea279ca878408707' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 6e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.06 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.050577 seconds (Warm-up)
## Chain 3:                0.037458 seconds (Sampling)
## Chain 3:                0.088035 seconds (Total)
## Chain 3:
##
```

```
## SAMPLING FOR MODEL '736af21cee6aa08cea279ca878408707' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 6e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.06 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.051211 seconds (Warm-up)
## Chain 4:                0.044847 seconds (Sampling)
## Chain 4:                0.096058 seconds (Total)
## Chain 4:
```

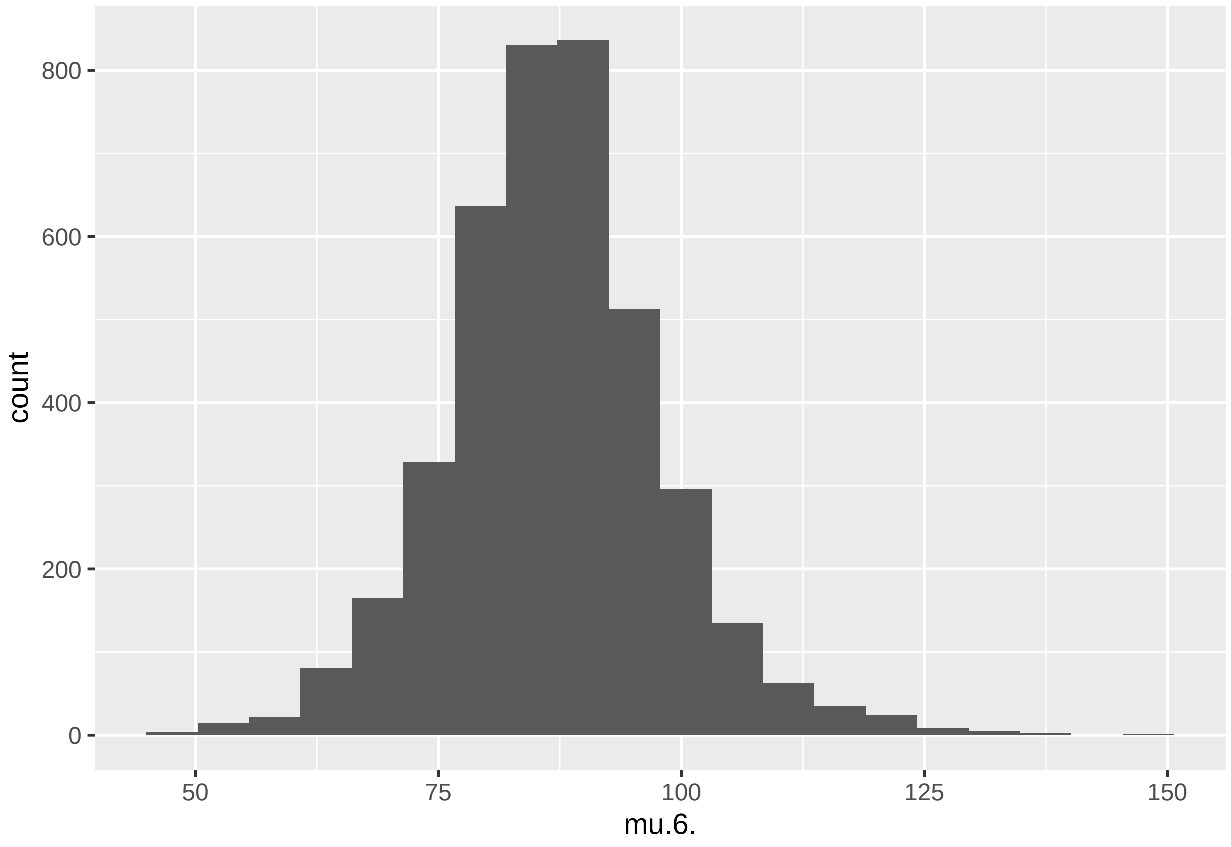
```
monitor(sm)
```

```
## Inference for the input samples (4 chains: each with iter = 2000; warmup = 0):
##
##             Q5   Q50   Q95  Mean   SD  Rhat Bulk_ESS Tail_ESS
## mu[1]     59.1  76.9  97.0  77.4 11.6     1     1932     1613
## mu[2]     92.7 106.0 119.1 105.9  8.8     1     1886     1260
## mu[3]     75.7  88.1 102.1  88.3  8.4     1     1778     1301
## mu[4]    102.8 111.4 119.7 111.3  5.5     1     1912     1185
## mu[5]     78.6  90.0 102.0  90.1  7.4     1     2303     1754
## mu[6]     69.0  86.7 105.7  87.0 11.2     1     2521     1916
## sigma[1]  13.4  22.6  46.5  25.3 11.0     1     2363     2230
## sigma[2]   8.4  14.6  32.6  16.8  8.6     1     1956     1404
## sigma[3]   9.0  15.4  32.9  17.5  8.4     1     2564     1899
## sigma[4]   5.3   9.5  22.4  11.1  5.8     1     2146     1956
## sigma[5]   7.6  13.3  29.4  15.3  7.9     1     2871     2213
## sigma[6]  13.0  22.3  45.5  25.0 11.2     1     2546     2330
## ypred[1]  30.4  76.2 124.4  76.6 29.5     1     3522     3359
## ypred[2]  72.5 106.1 139.7 106.2 21.6     1     3425     2860
## ypred[3]  56.8  88.7 121.4  88.9 20.8     1     3634     3359
## ypred[4]  90.5 111.8 133.1 111.7 13.3     1     3602     2702
## ypred[5]  62.0  90.2 117.6  90.0 18.7     1     3670     3293
## ypred[6]  40.1  86.6 132.4  86.8 29.3     1     3893     3592
## lp__     -59.7 -54.1 -50.3 -54.5  2.9     1     1173     1518
##
## For each parameter, Bulk_ESS and Tail_ESS are crude measures of
## effective sample size for bulk and tail quantities respectively (an ESS > 100
## per chain is considered good), and Rhat is the potential scale reduction
## factor on rank normalized split chains (at convergence, Rhat <= 1.05).
```

Plot the histograms for separate model:

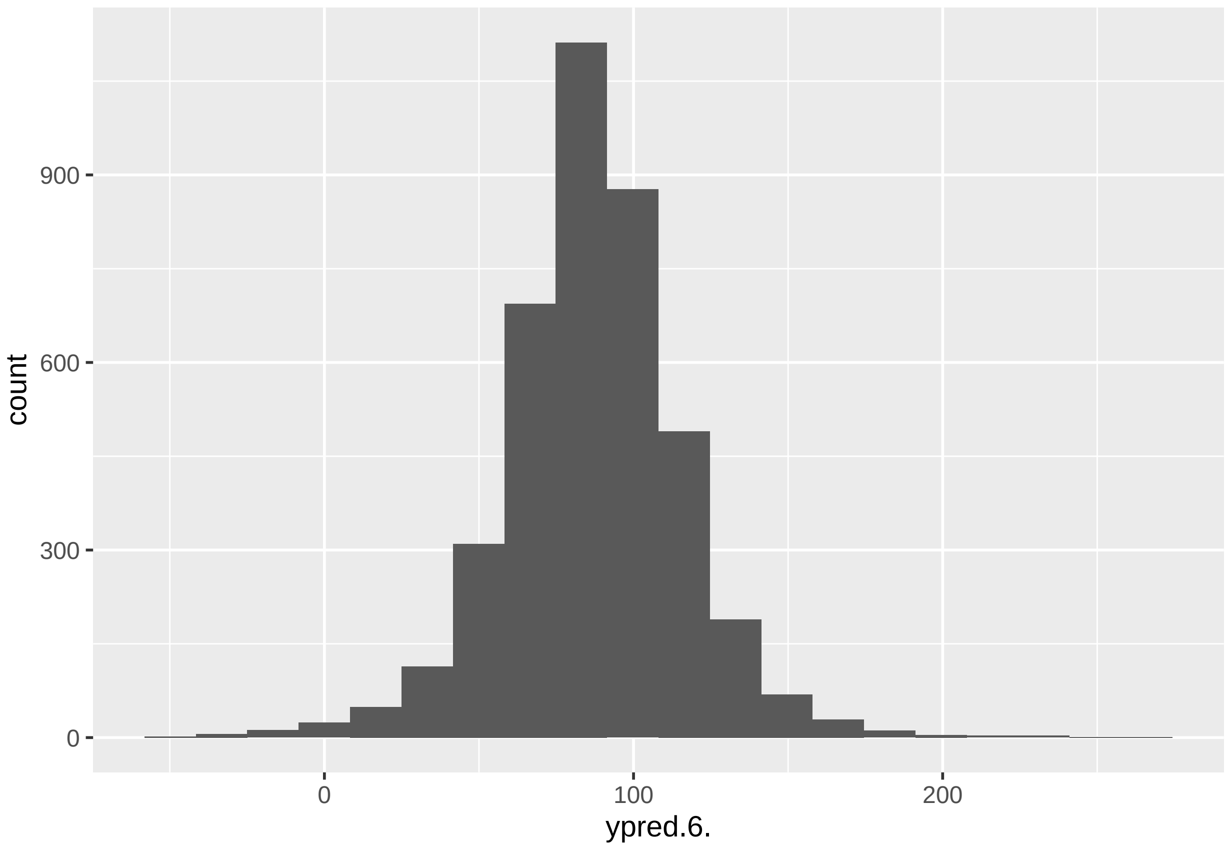
i) the posterior distribution of the mean of the quality measurements of the sixth machine.

```
draws <- as.data.frame(extract(sm, pars = c("mu[6]", "ypred[6]"), permuted = T))
posterior_mean <- ggplot(draws, aes(mu.6.)) + geom_histogram(bins=20)
posterior_mean
```



ii) the predictive distribution for another quality measurement of the sixth machine.

```
predictive_dist <- ggplot(draws, aes(ypred.6.)) + geom_histogram(bins=20)
predictive_dist
```



The predictive distribution is wider than the posterior distribution for the 6th machine.

iii) the posterior distribution of the mean of the quality measurements of the seventh machine.

We cannot plot a histogram of the posterior distribution for the 7th machine, since we are modeling each machine separately and cannot say anything about seventh machine based on these models.

The posterior expectation for μ_1 with a 90% credibility interval for separate model but using a normal(0, 10) prior for the μ parameter(s) and a Gamma(1, 1) prior for the σ parameter(s).

SEPARATE MODEL WITH STAN FOR CREDIBLE INTERVAL

```
data {
  int < lower =0 > N; // n of measurements
  int < lower =0 > J; // n of machines
  vector[J] y[N];
}
parameters {
  vector<lower = 0>[J] mu ;
  vector<lower = 0>[J] sigma ;
}
model {
  // weakly informative priors
  for ( j in 1: J ){
    mu [j] ~ normal (0, 10);
    sigma [j] ~ gamma(1, 1);
  }
  // likelihood
  for ( j in 1: J )
    y[ ,j ] ~ normal (mu[j], sigma[j]);
}
```

```
fit_sm <- sampling(separate_model_cred, data = stan_data, refresh = 0)
# 90% credible interval
monitor(fit_sm, probs = c(0.05, 0.95), print=FALSE)[c(1), c(4,5)]
```

```
##             5%      95%
## mu[1] 34.98894 64.39992
```

Pooled model

In the pooled model, all the measurements are combined and no distinction is made between the machines (they are drawn from the same distribution and the paramters do not change between the machines). We are using the same weakly informative priors as earlier in the separate model as there is no need to change them.

Pooled model described with mathematical notation

$$y_{ij} \sim N(\mu, \sigma)$$
$$\mu \sim N(100, 50)$$
$$\sigma \sim Cauchy(0, 40) > 0$$

POOLED MODEL WITH STAN

```
data {
  int < lower =0 > N; // n of measurements
  int < lower =0 > J; // n of machines
  vector[J] y[N];
}
parameters {
  real<lower = 0> mu ;
  real<lower = 0> sigma ;
}
model {
  // weakly informative priors
  mu ~ normal (100, 50);
  sigma ~ cauchy(0, 40);
  // likelihood
  for ( j in 1: J )
    y[ ,j ] ~ normal (mu, sigma);
}
generated quantities {
  real ypred ;
  // Compute predictive distribution for a machine as we cannot tell the
  // difference between the machines in the pooled model.
  ypred = normal_rng(mu, sigma);
}
```

```r
stan_data <- list(y = factory,N = nrow(factory),J = ncol(factory))
pm <- rstan::sampling(pooled_model, data = stan_data)
```

```
##
## SAMPLING FOR MODEL '4712b8e1e3c79779aa0e0ea518879bee' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 5e-06 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.05 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.014162 seconds (Warm-up)
## Chain 1:                0.011759 seconds (Sampling)
## Chain 1:                0.025921 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '4712b8e1e3c79779aa0e0ea518879bee' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 4e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.04 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.013842 seconds (Warm-up)
## Chain 2:                0.014601 seconds (Sampling)
## Chain 2:                0.028443 seconds (Total)
## Chain 2:
##
```

```
## SAMPLING FOR MODEL '4712b8e1e3c79779aa0e0ea518879bee' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 4e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.04 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.013693 seconds (Warm-up)
## Chain 3:                0.011411 seconds (Sampling)
## Chain 3:                0.025104 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '4712b8e1e3c79779aa0e0ea518879bee' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 4e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.04 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.013555 seconds (Warm-up)
## Chain 4:                0.013779 seconds (Sampling)
## Chain 4:                0.027334 seconds (Total)
## Chain 4:
```

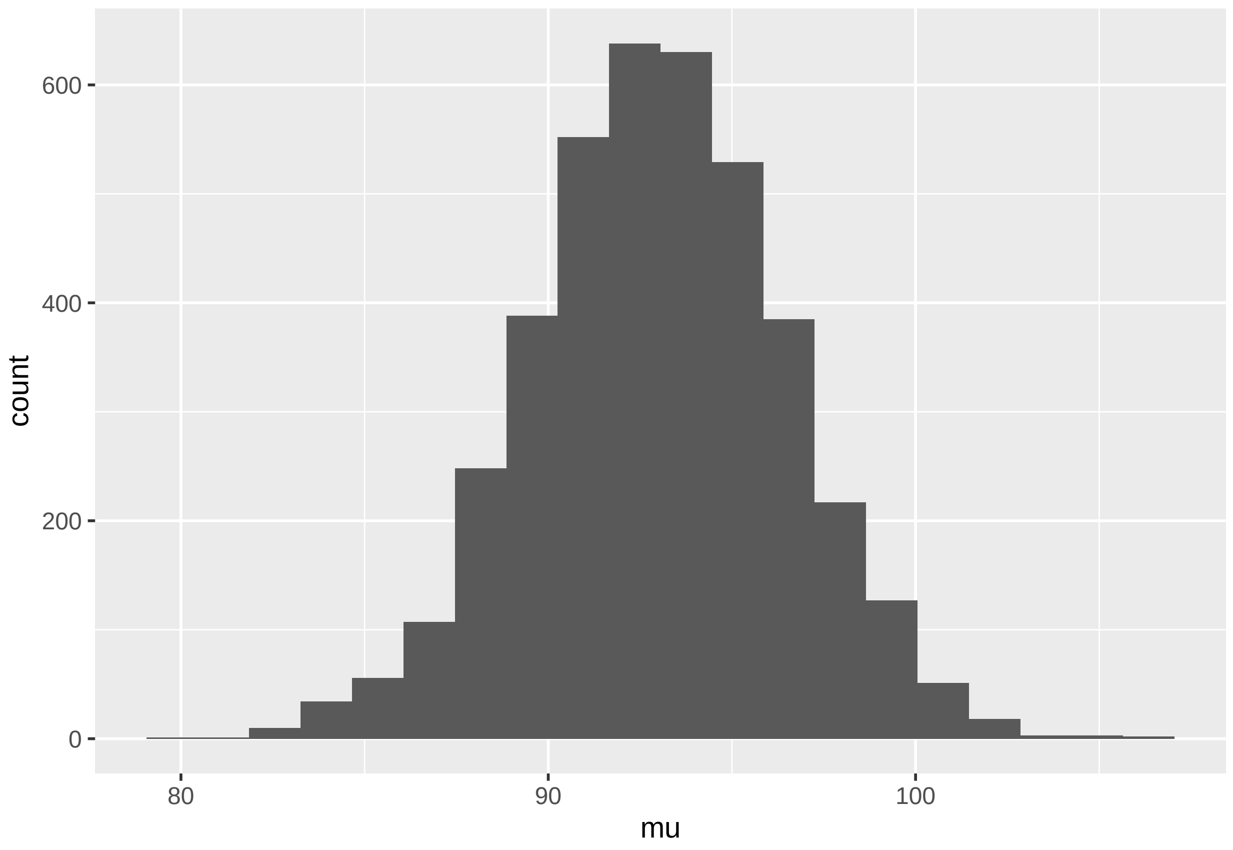
```r
monitor(pm)
```

```
## Inference for the input samples (4 chains: each with iter = 2000; warmup = 0):
##
##          Q5   Q50   Q95  Mean   SD  Rhat Bulk_ESS Tail_ESS
## mu      87.4  93.0  98.7  93.0  3.5     1     2533     2229
## sigma   15.1  18.5  23.3  18.7  2.5     1     3072     2611
## ypred   61.7  92.7 123.5  93.0 19.0     1     4087     3924
## lp__   -97.0 -94.7 -94.0 -95.0  1.0     1     1843     2580
##
## For each parameter, Bulk_ESS and Tail_ESS are crude measures of
## effective sample size for bulk and tail quantities respectively (an ESS > 100
## per chain is considered good), and Rhat is the potential scale reduction
## factor on rank normalized split chains (at convergence, Rhat <= 1.05).
```

Plot the histograms for the pooled model:

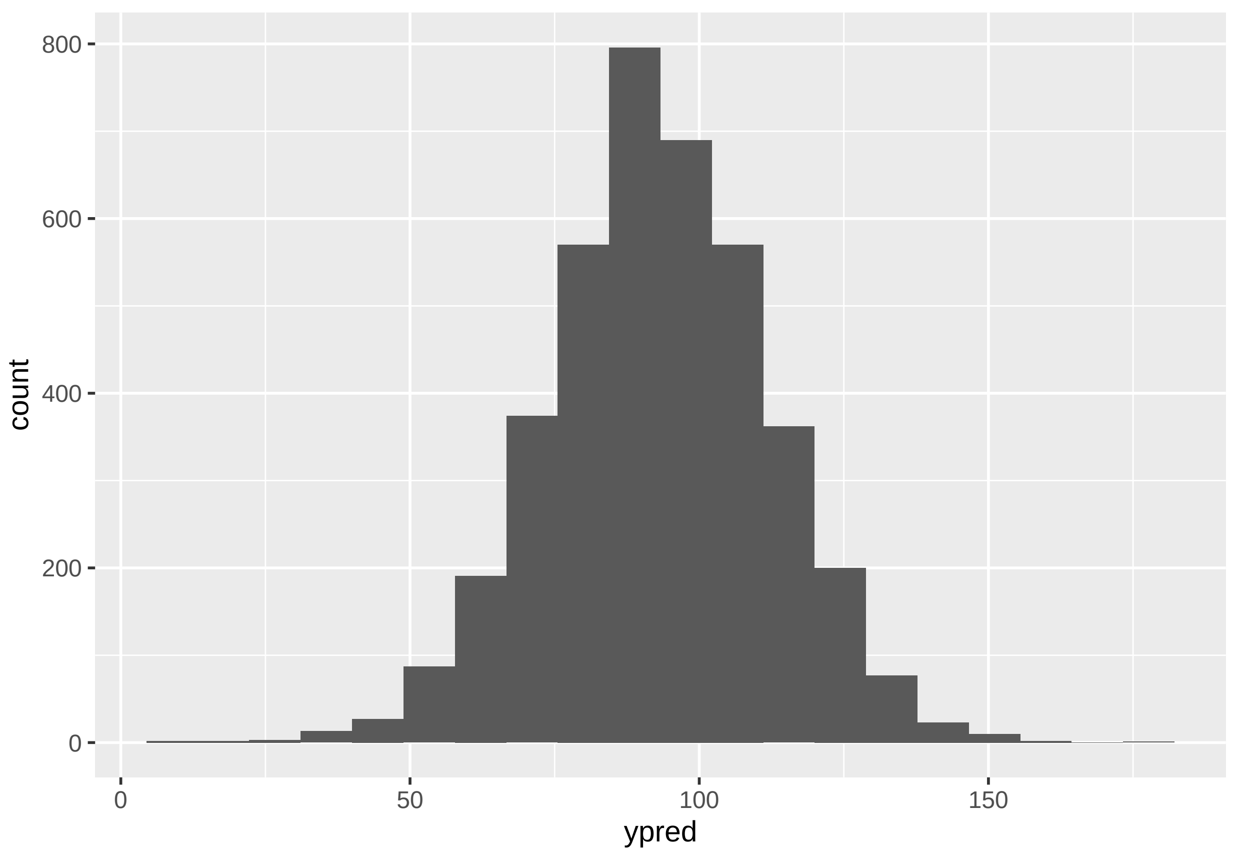
i) the posterior distribution of the mean of the quality measurements of the sixth machine.

```r
draws <- as.data.frame(extract(pm, pars = c("mu", "ypred"), permuted = T))
posterior_mean <- ggplot(draws, aes(mu)) + geom_histogram(bins=20)
posterior_mean
```



ii) the predictive distribution for another quality measurement of the sixth machine.

```r
predictive_dist <- ggplot(draws, aes(ypred)) + geom_histogram(bins=20)
predictive_dist
```



iii) the posterior distribution of the mean of the quality measurements of the seventh machine.

When modeling with pooled model, the distribution of the mean quality measurement of the nth machine is the exact same as for any of the 1-6 machines. We would assume that the histogram follows the same distribution as visualized in the first plot. So, to answer the question directly, it would look similar to the first plot.

The posterior expectation for μ_1 with a 90% credibility interval for pooled model but using a normal(0, 10) prior for the μ parameter(s) and a Gamma(1, 1) prior for the σ parameter(s).

POOLED MODEL WITH STAN FOR CREDIBILITY INTERVAL

```
data {
  int < lower =0 > N; // n of measurements
  int < lower =0 > J; // n of machines
  vector[J] y[N];
}
parameters {
  real<lower = 0> mu ;
  real<lower = 0> sigma ;
}
model {
  // weakly informative priors
  mu ~ normal (0, 10);
  sigma ~ gamma(1, 1);
  // likelihood
  for ( j in 1: J )
    y[ ,j ] ~ normal (mu, sigma);
}
```

```
fit_pm <- sampling(pooled_model_cred, data = stan_data, refresh = 0)
# 90% credible interval
monitor(fit_pm, probs = c(0.05, 0.95), print=FALSE)[c(1), c(4,5)]
```

```
##          5%      95%
## mu 80.24915 90.25796
```

Hierarchical model

In hierarchical model, as in the model described in the book, use the same measurement standard deviation σ for all the groups in the hierarchical model. Again, there is no need to change our priors for this case. Only thing changing is that we need to use the hyper-priors now in addition to priors.

Hierarchical model described with mathematical notation

$$y_j \sim N(\theta_{ij}, \sigma)$$
$$\theta_j \sim N(\mu, \tau)$$
$$\sigma \sim Cauchy(0, 40) > 0$$

Hyper-priors

$$\mu \sim N(100, 50)$$
$$\tau \sim Cauchy(0, 40) > 0$$

HIERARCHICAL MODEL WITH STAN

```
data {
  int < lower =0 > N; // n of measurements
  int < lower =0 > J; // n of machines
  vector[J] y[N];
}
parameters {
  real mu; // hyper-parameter 1
  real<lower=0> tau; // hyper-parameter 2
  vector[J] theta; // separate mean parameter theta for each machine
  real<lower=0> sigma; // common sigma parameter for all machines
}
model {
  // weakly informative priors
  mu ~ normal (100, 50); // hyperprior for mu
  tau ~ cauchy(0, 40); // hyperprior for tau
  for ( j in 1: J ){
    theta [j] ~ normal (mu, tau);
  }
  sigma ~ cauchy(0,40);
  // likelihood
  for ( j in 1: J )
    y[ ,j ] ~ normal (theta[j], sigma);
}
generated quantities {
  vector[J] ypred ;
  real theta7;

  for (j in 1:J)
    ypred[j] = normal_rng(theta[j], sigma);
  theta7 = normal_rng(mu, tau);
}
```

```r
stan_data <- list(y = factory,N = nrow(factory),J = ncol(factory))
hm <- rstan::sampling(hierarchical_model, data = stan_data)
```

```
##
## SAMPLING FOR MODEL '4b1bc4443793bae3b458c82a47cba3e1' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 8e-06 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.076995 seconds (Warm-up)
## Chain 1:                0.031038 seconds (Sampling)
## Chain 1:                0.108033 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '4b1bc4443793bae3b458c82a47cba3e1' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 5e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.05 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.067042 seconds (Warm-up)
## Chain 2:                0.033963 seconds (Sampling)
## Chain 2:                0.101005 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '4b1bc4443793bae3b458c82a47cba3e1' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 5e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.05 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.081956 seconds (Warm-up)
## Chain 3:                0.026828 seconds (Sampling)
## Chain 3:                0.108784 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '4b1bc4443793bae3b458c82a47cba3e1' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 5e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.05 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.085564 seconds (Warm-up)
## Chain 4:                0.03188 seconds (Sampling)
## Chain 4:                0.117444 seconds (Total)
## Chain 4:
```

```
## Warning: There were 56 divergent transitions after warmup. See
## http://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
```

```
## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quant
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#tail-ess
```

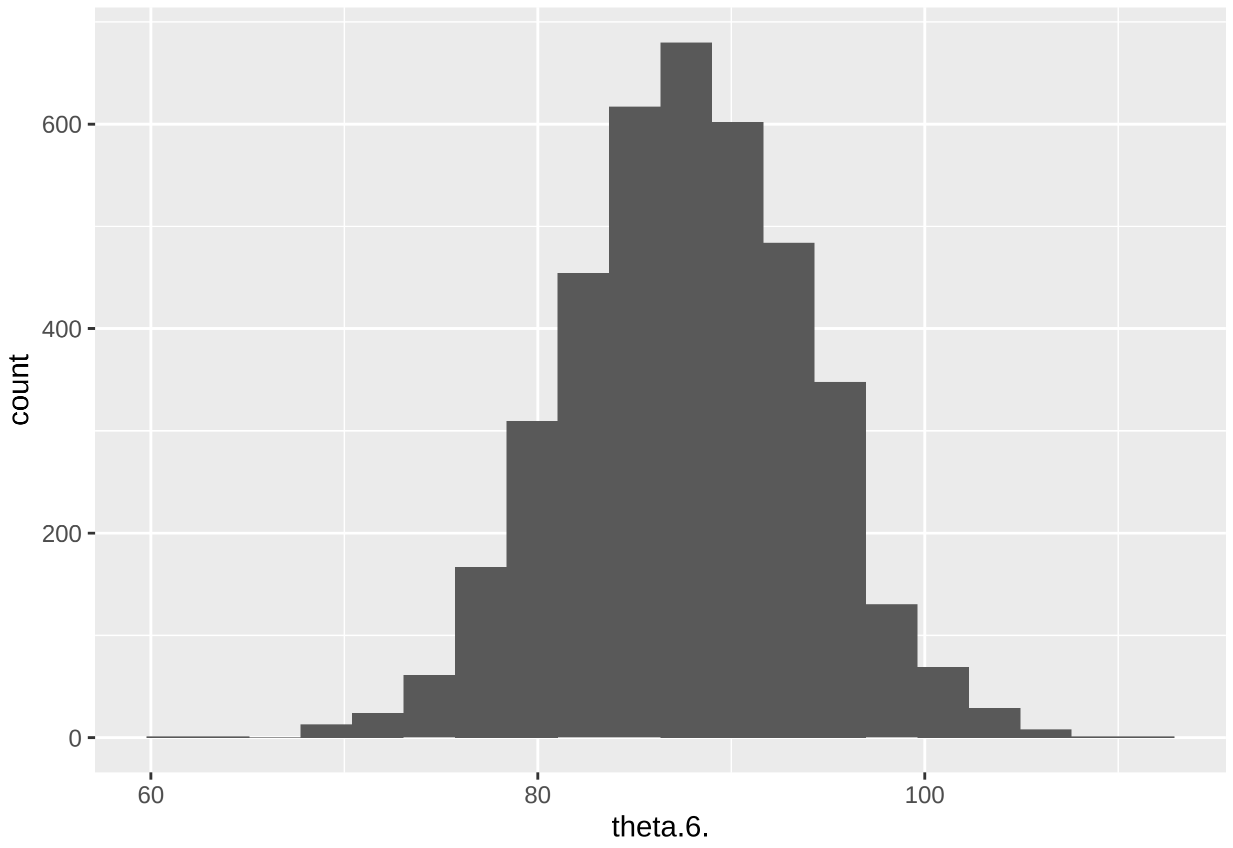
```r
monitor(hm)
```

```
## Inference for the input samples (4 chains: each with iter = 2000; warmup = 0):
##
##               Q5    Q50    Q95   Mean   SD  Rhat Bulk_ESS Tail_ESS
## mu          81.4   92.9  104.2   92.9  7.1  1.00     2664     2035
## tau          5.1   13.4   29.5   14.8  8.1  1.01      401      120
## theta[1]    69.2   80.2   92.7   80.5  7.0  1.01      655      211
## theta[2]    91.9  102.8  113.7  102.8  6.7  1.00     2027     1652
## theta[3]    79.2   89.3   98.9   89.2  6.1  1.00     2682     2733
## theta[4]    94.4  107.2  118.5  106.9  7.2  1.00      756      953
## theta[5]    80.9   90.7  100.1   90.6  5.9  1.00     3210     2745
## theta[6]    77.5   87.7   97.5   87.7  6.2  1.00     2329     2693
## sigma       11.9   14.9   19.8   15.3  2.4  1.00     1145     1570
## ypred[1]    53.5   80.5  107.5   80.4 16.6  1.00     3028     3655
## ypred[2]    75.5  103.9  130.4  103.2 16.7  1.00     3501     3692
## ypred[3]    62.9   88.9  116.1   89.3 16.5  1.00     3859     3887
## ypred[4]    78.3  107.5  134.7  107.3 17.2  1.00     2886     3376
## ypred[5]    63.2   90.7  117.5   90.6 16.7  1.00     3809     3825
## ypred[6]    61.2   88.1  115.2   87.9 16.5  1.00     3697     3714
## theta7      64.5   93.1  122.3   93.4 18.6  1.00     3867     3369
## lp__      -113.7 -108.6 -105.6 -109.0  2.5  1.01      446      154
##
## For each parameter, Bulk_ESS and Tail_ESS are crude measures of
## effective sample size for bulk and tail quantities respectively (an ESS > 100
## per chain is considered good), and Rhat is the potential scale reduction
## factor on rank normalized split chains (at convergence, Rhat <= 1.05).
```

Plot the histograms for the hierarchical model:

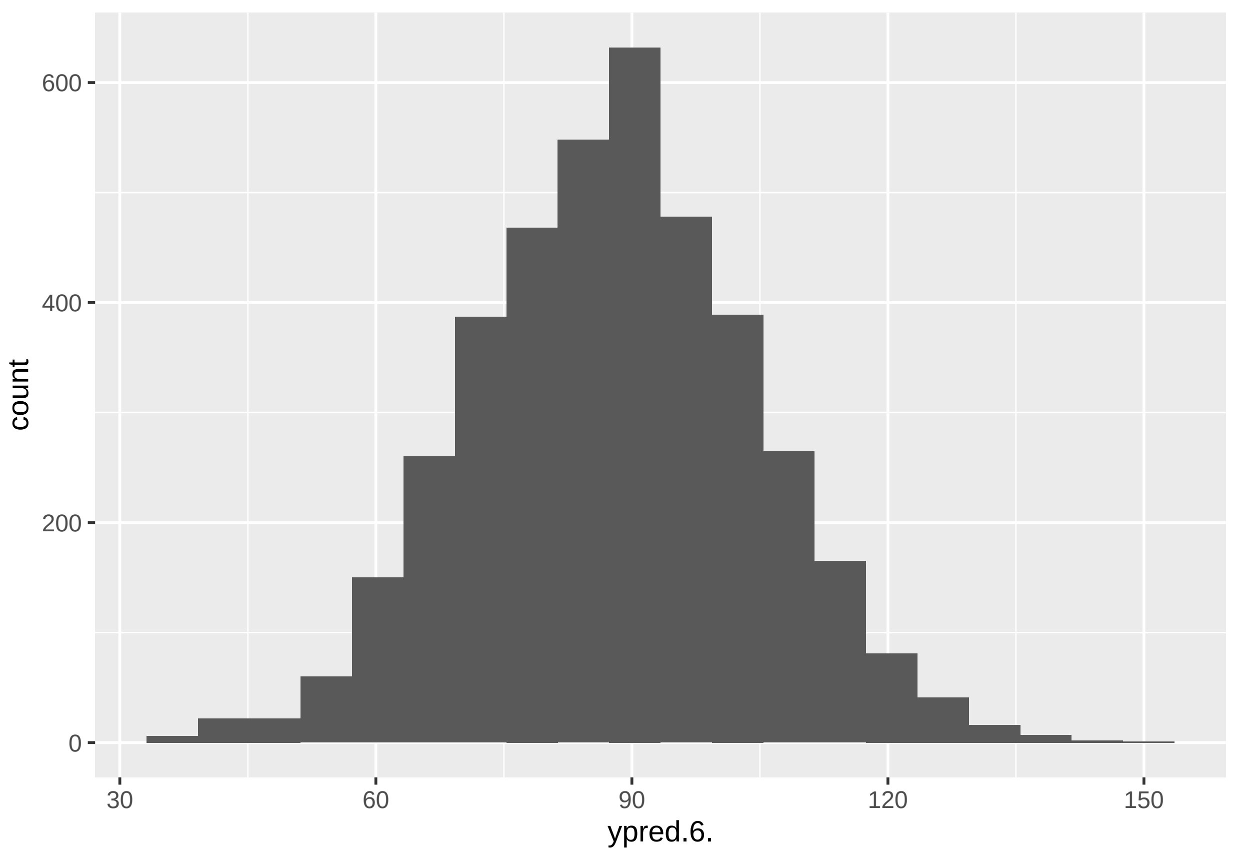
i) the posterior distribution of the mean of the quality measurements of the sixth machine.

```r
draws <- as.data.frame(extract(hm, pars = c("theta[6]", "ypred[6]", "theta7"), permuted = T))
posterior_mean <- ggplot(draws, aes(theta.6.)) + geom_histogram(bins=20)
posterior_mean
```



ii) the predictive distribution for another quality measurement of the sixth machine.

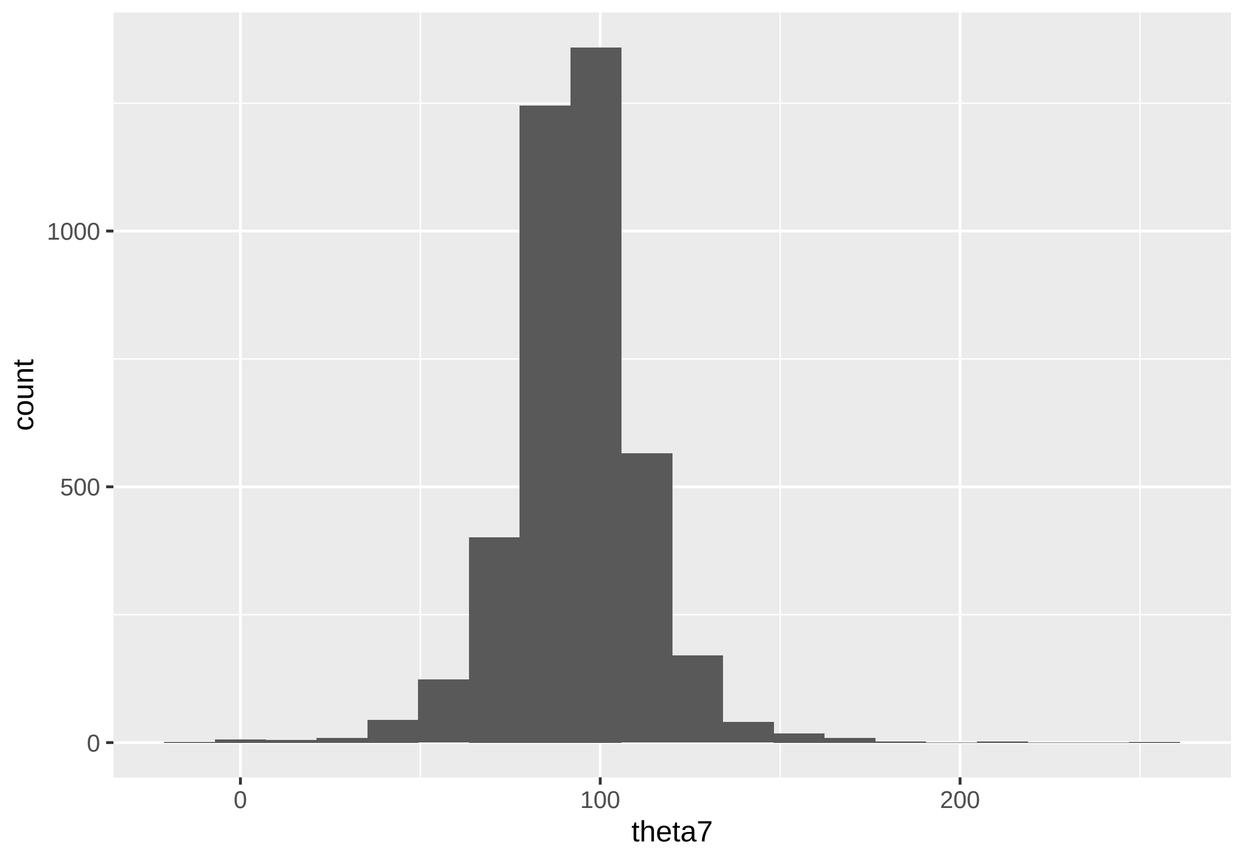
```
predictive_dist <- ggplot(draws, aes(ypred.6.)) + geom_histogram(bins=20)
predictive_dist
```



The predictive distribution is quite close to the posterior distribution. This means that our model predicts the future measurements pretty well compared to separate model where the predictive distribution was wider than here. This has to occur due to the fact that in hierarchical model we take into account information also from other machines when we try to predict for the future measurement of sixth machine. In separate model we only predict based on one machine.

iii) the posterior distribution of the mean of the quality measurements of the seventh machine.

```
predictive_dist <- ggplot(draws, aes(theta7)) + geom_histogram(bins=20)
predictive_dist
```



The posterior expectation for μ_1 with a 90% credibility interval for hierarchical model but using a normal(0, 10) hyper-prior for the μ parameter(s) and a Gamma(1, 1) hyper-prior for the σ parameter(s).

HIERARCHICAL MODEL WITH STAN FOR CREDIBILITY INTERVAL

```
data {
  int < lower =0 > N; // n of measurements
  int < lower =0 > J; // n of machines
  vector[J] y[N];
}
parameters {
  real mu; // hyper-parameter 1
  real<lower=0> tau; // hyper-parameter 2
  vector[J] theta; // separate mean parameter theta for each machine
  real<lower=0> sigma; // common sigma parameter for all machines
}
model {
  // weakly informative priors
  mu ~ normal (0, 10); // hyperprior for mu
  tau ~ gamma(1, 1); // hyperprior for tau
  for ( j in 1: J ){
    theta [j] ~ normal (mu, tau);
  }
  sigma ~ gamma(1, 1);
  // likelihood
  for ( j in 1: J )
    y[ ,j ] ~ normal (theta[j], sigma);
}
```

```r
fit_hm <- sampling(hierarchical_model_cred, data = stan_data, refresh = 0)
```

```
## Warning: There were 77 divergent transitions after warmup. See
## http://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

## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#bulk-ess

## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quant
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#tail-ess
```

```r
# 90% credible interval
monitor(fit_hm, probs = c(0.05, 0.95), print=FALSE)[c(3), c(4,5)]
```

```
##                 5%      95%
## theta[1] 66.97701 84.55162
```