BDA - Assignment 7

Anonymous

Exercise 1

Fix 1: The upper bound of sigma was set to 0. Sigma cannot be negative, therefore I corrected the "real<upper=0>" to "real<lower=0>".

Fix 2: In the last block (generated quantities), the ypred was drawn from wrongly parametrized distribution. To fix this, the parameter mu was switched to alpha + beta*xpred, which corresponds to the correct case.

Below is the full modified Stan model script.

```
# STAN CODE
writeLines(readLines("drowning_lm.stan"))
```

```
## data {
##   int<lower=0> N;
##   vector[N] x;
##   vector[N] y;
##   real xpred;
##   real pmubeta;
##   real psbeta;
## }
## parameters {
##   real alpha;
##   real beta;
##   real<lower=0> sigma;
## }
## transformed parameters {
##   vector[N] mu;
##   mu = alpha + beta*x;
## }
## model {
##   beta ~ normal(pmubeta, psbeta);
##   y ~ normal(mu, sigma);
## }
## generated quantities {
##   real ypred;
##   ypred = normal_rng(alpha + beta*xpred, sigma);
## }
```

Our prior for the slope is $N(0, \tau^2)$. We wanted to pick τ in a way where $P(-69 < \beta < 69) = 0.99$ holds.

```
sd_candidates = seq(0, 40)
lower_bound <- qnorm(0.005, 0, sd_candidates)
```

We see that the correct deviation is somewhere around 26. We notice that $\tau = 26.8$ is a reasonable numerical value as:

```r
lower_bound = qnorm(0.005, 0, 26.8)
lower_bound
```

```
## [1] -69.03223
```

```r
# SOURCE CODE

# Load the data
data("drowning")

# Combine the data
data_to_fit <- list(N = length(drowning$drownings), x=drowning$year, y = drowning$drownings, xpred = 20

# Run the stan file
stan_object <- stan("drowning_lm.stan", data=data_to_fit)
```

```
##
## SAMPLING FOR MODEL 'drowning_lm' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 2.611 seconds (Warm-up)
## Chain 1:                3.891 seconds (Sampling)
## Chain 1:                6.502 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'drowning_lm' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 2.469 seconds (Warm-up)
## Chain 2:                3.839 seconds (Sampling)
## Chain 2:                6.308 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'drowning_lm' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 2.246 seconds (Warm-up)
## Chain 3:                4.206 seconds (Sampling)
## Chain 3:                6.452 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'drowning_lm' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 1.981 seconds (Warm-up)
## Chain 4:                4.391 seconds (Sampling)
## Chain 4:                6.372 seconds (Total)
## Chain 4:
```

```
## Warning: There were 1182 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
```

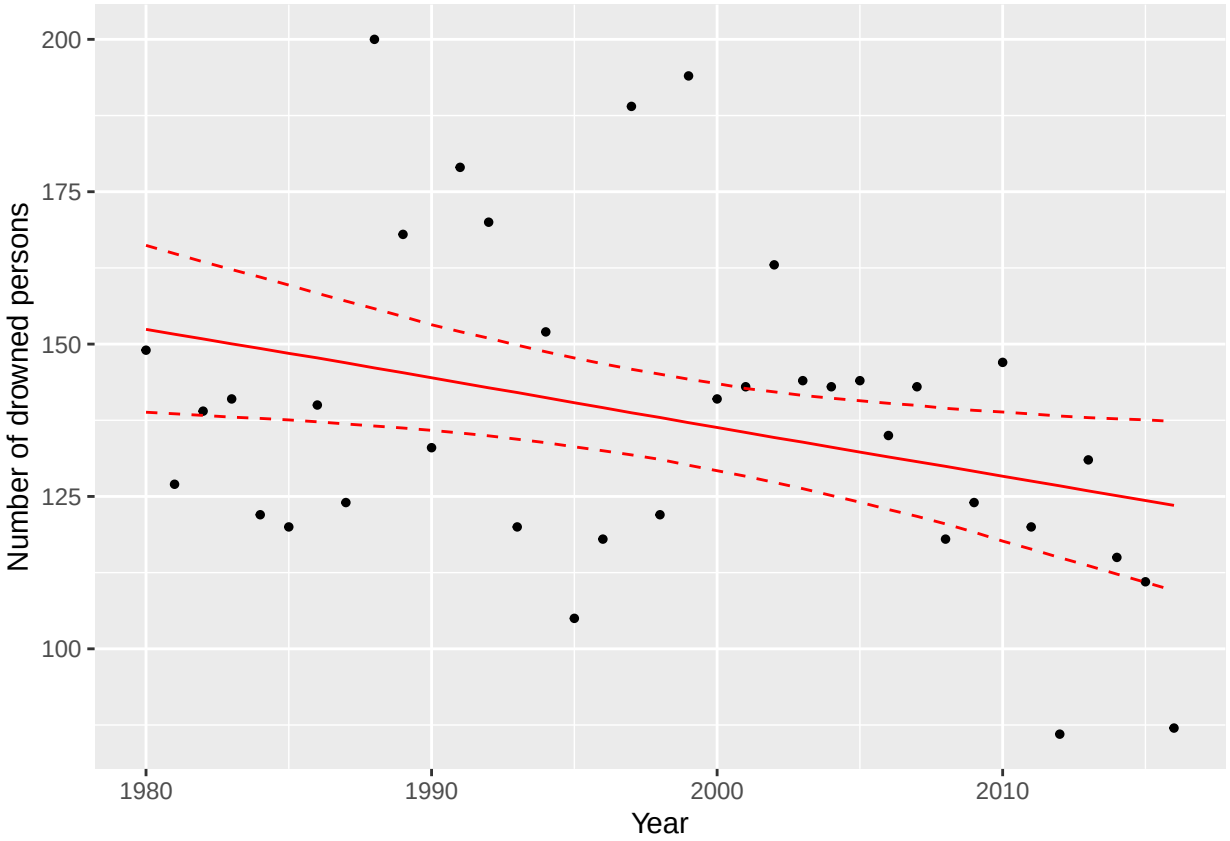
I implemented the desired prior followingly:

- Data block: I added two new values. The parameters of the prior distribution of the slope (real pmubeta, real psbeta), namely (0, 26.8).
- Model block: I added the prior distribution of the beta: "beta ~ normal(pmubeta, psbeta);"

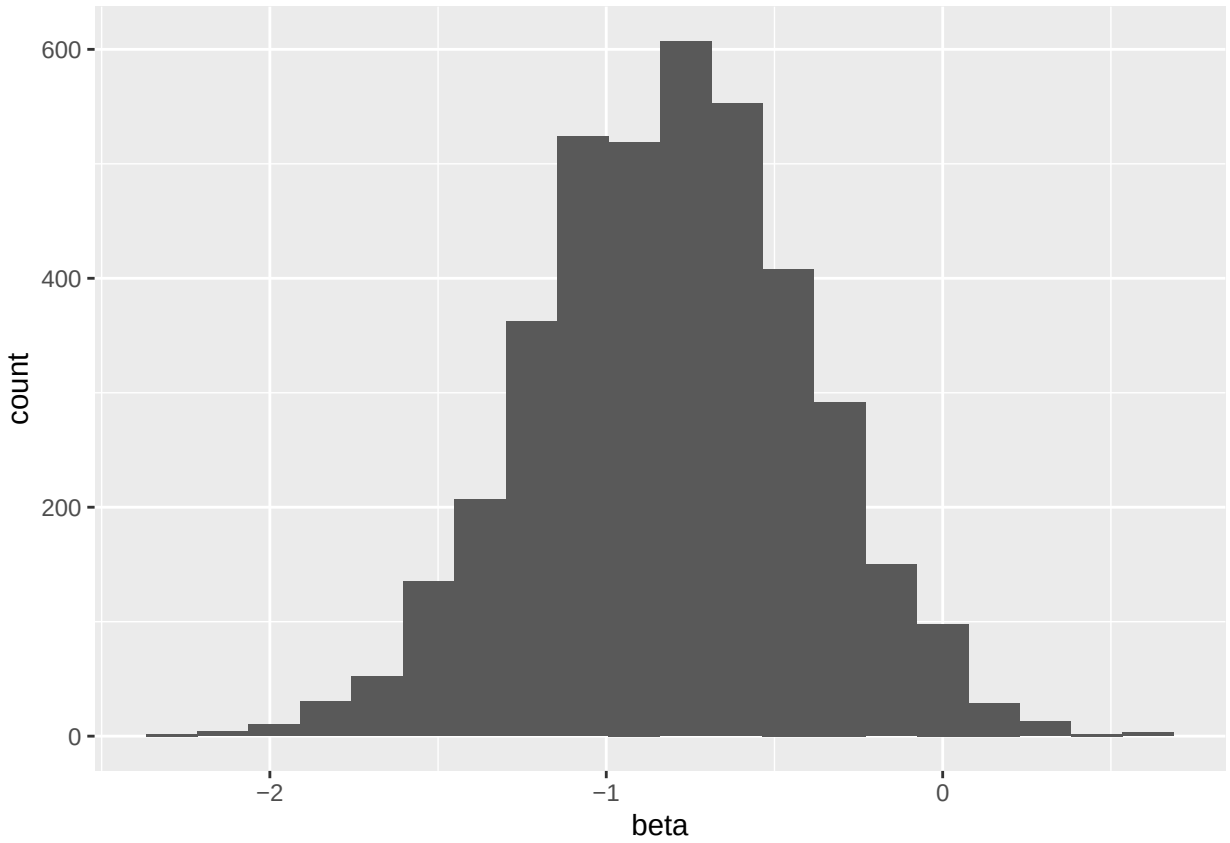
Finally we check the results if they seem correct.

```
df = data.frame(x = drowning$year, y = drowning$drownings)
draws_mu <- extract(stan_object, pars = "mu", permuted=TRUE)

mu <- apply(draws_mu$mu, 2, quantile, c(0.05, 0.5, 0.95)) %>%
  t() %>% data.frame(x = df$x, .)

colnames(mu) <- c("year", "x5", "x50", "x95")

trend_plot <- ggplot() +
  geom_point(aes(x, y), data = df, size = 1) +
  labs(y = "Number of drowned persons", x= "Year") +
  guides(linetype = F) + geom_line(aes(year, x50), data = mu, color = 'red') +
  geom_line(aes(year, x5), data = mu, color = 'red', linetype = "dashed") +
  geom_line(aes(year, x95), data = mu, color = 'red', linetype = "dashed")


all_draws = as.data.frame(extract(stan_object, permuted = TRUE))

hist_slope <- ggplot(all_draws, aes(beta)) + geom_histogram(bins=20)
hist_pred <- ggplot(all_draws, aes(ypred)) + geom_histogram(bins=20)
```

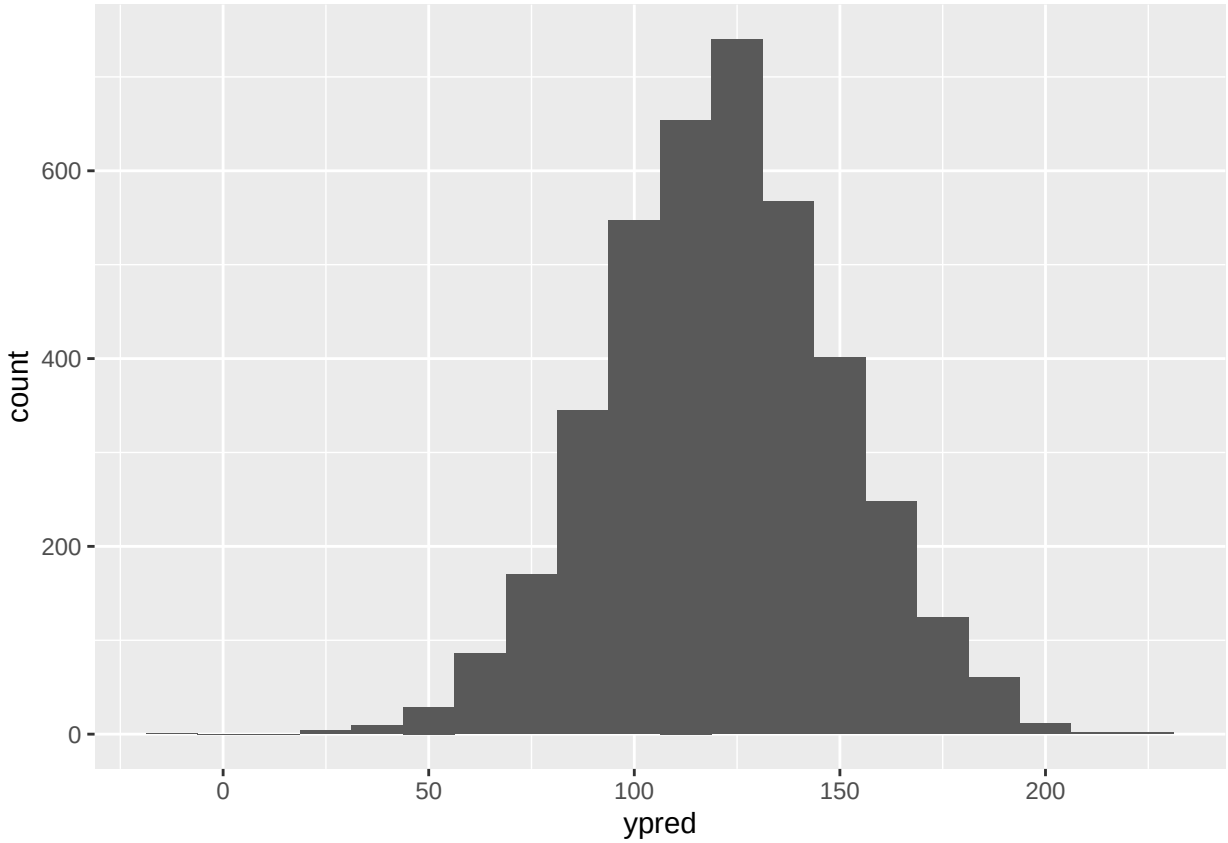
```
trend_plot
```


200
175
150
125
100
Number of drowned persons
1980
1990
2000
2010
Year

```
hist_slope
```




```
hist_pred
```



The results seem reliable as they are very similar to ones in the assignment.

EXERCISE 2

SEPARATE MODEL

```
data("factory")
```

```
# STAN CODE
writeLines(readLines("model_separate.stan"))
```

```
## data {
##   int<lower=0> N;
##   int<lower=0> K;
##   vector[N] y[K];
## }
## parameters {
##   vector[K] mu;
##   vector<lower=0>[K] sigma;
## }
## model {
##   for (j in 1:K)
##     y[j] ~ normal(mu[j], sigma[j]);
## }
```

```
## generated quantities {
##   vector[K] ypred;
##   for (i in 1:K)
##     ypred[i] = normal_rng(mu[i], sigma[i]);
## }
```

```
N <- 5
K <- 6
y <- t(factory)
data_separate <- list(N = N, K = K, y = y)

separate_result <- stan("model_separate.stan", data=data_separate)
```

```
##
## SAMPLING FOR MODEL 'model_separate' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.291 seconds (Warm-up)
## Chain 1:                0.073 seconds (Sampling)
## Chain 1:                0.364 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'model_separate' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.159 seconds (Warm-up)
## Chain 2:                0.061 seconds (Sampling)
## Chain 2:                0.22 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'model_separate' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.18 seconds (Warm-up)
## Chain 3:                0.081 seconds (Sampling)
## Chain 3:                0.261 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'model_separate' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0 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)
```

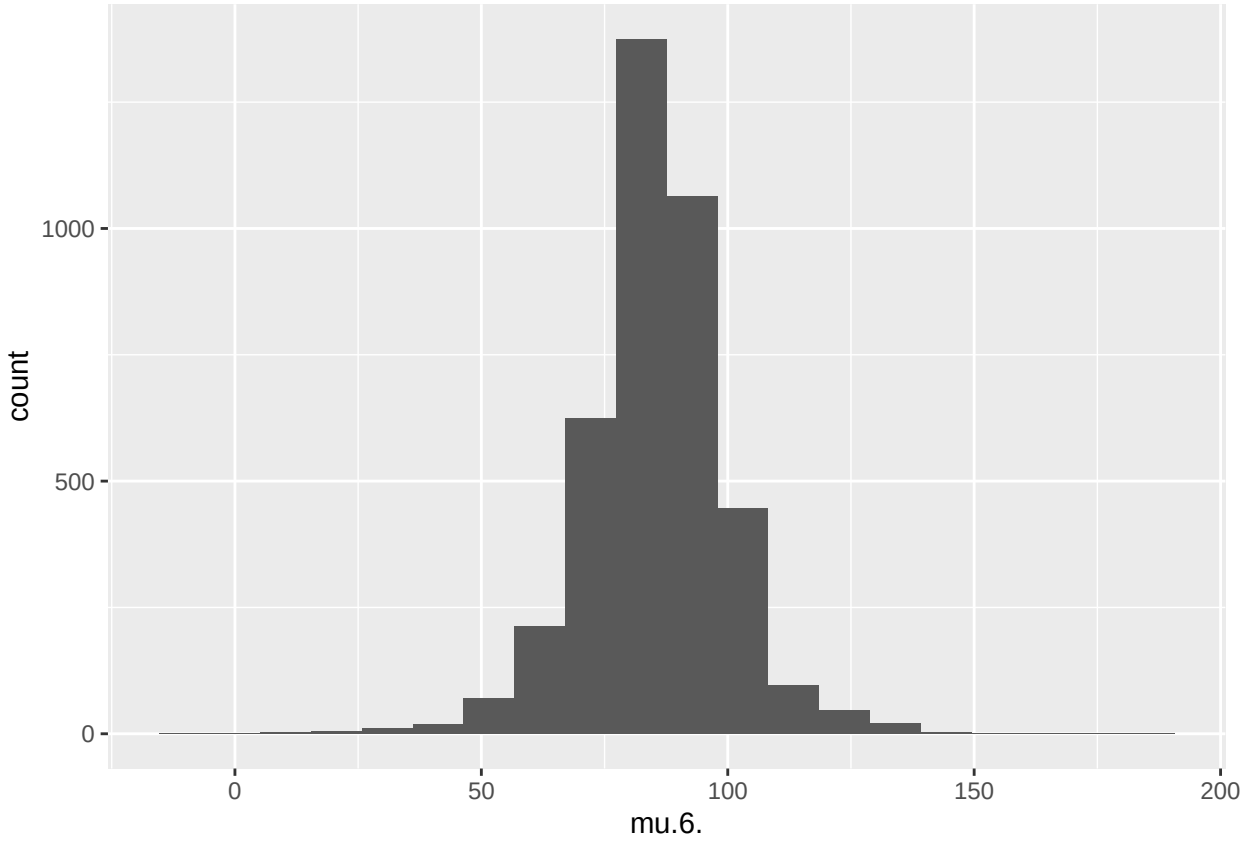
```
## Chain 4:
## Chain 4:  Elapsed Time: 0.187 seconds (Warm-up)
## Chain 4:                0.081 seconds (Sampling)
## Chain 4:                0.268 seconds (Total)
## Chain 4:
```

```
show(separate_result)
```

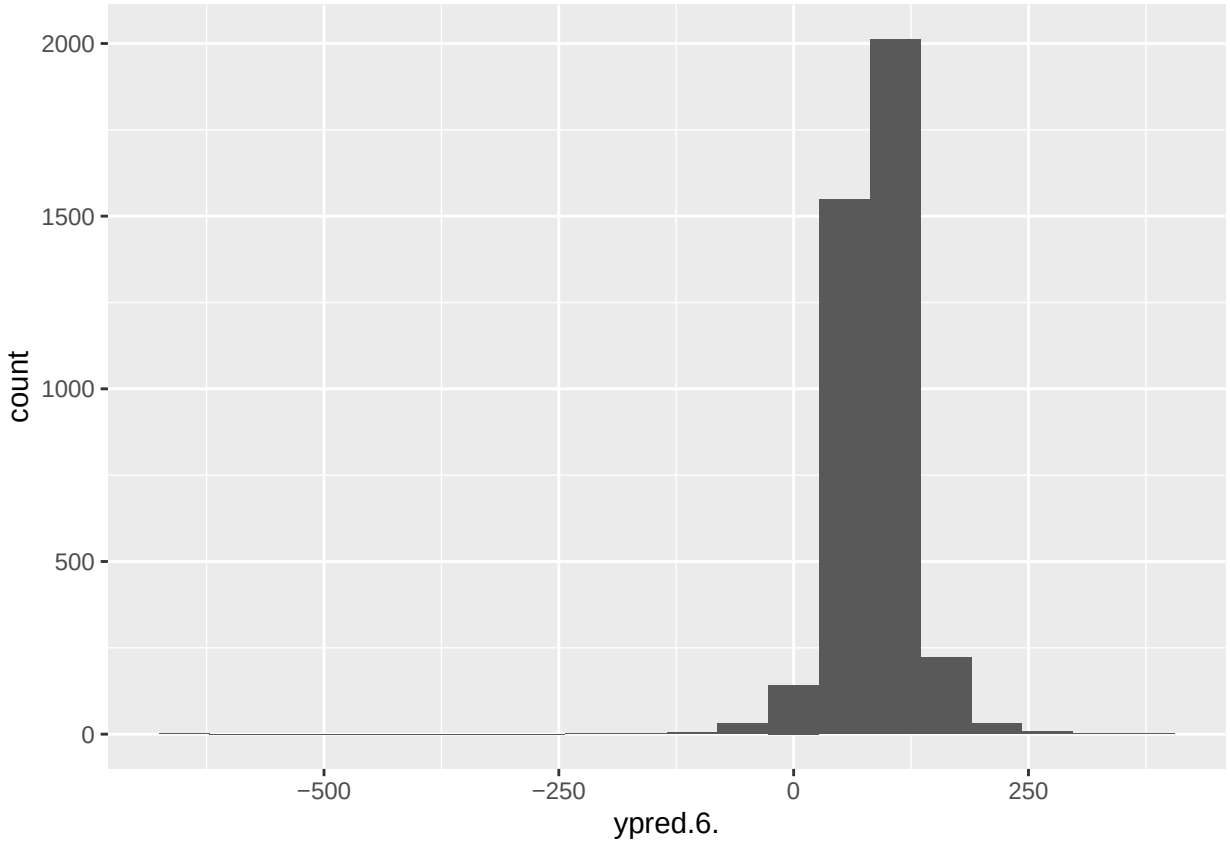
```
## Inference for Stan model: model_separate.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##            mean se_mean    sd   2.5%    25%    50%    75%  97.5% n_eff
## mu[1]     75.75    0.45 17.14  41.66  68.11  75.98  83.80 109.62  1436
## mu[2]    106.69    0.26  9.57  89.43 101.71 106.34 111.01 126.36  1347
## mu[3]     88.93    0.47 12.17  68.88  82.89  88.05  93.49 115.71   666
## mu[4]    111.52    0.16  5.91  99.66 108.51 111.61 114.56 123.54  1334
## mu[5]     89.85    0.21  8.38  71.67  85.81  89.97  94.21 106.41  1666
## mu[6]     85.31    0.33 14.26  55.70  77.88  85.56  93.12 113.67  1915
## sigma[1]  32.03    0.68 23.01  12.74  19.43  25.69  36.08  91.19  1147
## sigma[2]  18.67    0.40 13.16   7.82  11.54  15.16  21.23  52.44  1107
## sigma[3]  21.25    0.64 16.68   8.61  12.73  16.73  23.40  63.27   675
## sigma[4]  11.90    0.21  7.53   4.95   7.44   9.78  13.85  31.17  1239
## sigma[5]  16.59    0.34 10.66   7.19  10.56  13.69  19.09  44.11  1004
## sigma[6]  29.75    0.45 18.44  12.64  18.82  24.94  34.64  76.82  1648
## ypred[1]  76.38    0.84 43.82  -8.54  56.91  76.82  95.87 157.45  2726
## ypred[2] 106.48    0.49 26.20  60.26  95.61 106.09 117.24 153.86  2850
## ypred[3]  88.92    0.56 28.53  38.50  75.73  88.05 100.59 148.41  2598
## ypred[4] 111.40    0.26 15.37  81.80 104.26 111.42 118.83 141.61  3582
## ypred[5]  89.49    0.38 21.30  45.31  79.55  89.67  99.63 130.83  3105
## ypred[6]  85.39    0.63 38.86   7.21  67.10  85.52 104.08 161.59  3829
## lp__     -81.26    0.12  3.24 -88.68 -83.24 -80.82 -78.95 -76.12   767
##          Rhat
## mu[1]       1
## mu[2]       1
## mu[3]       1
## mu[4]       1
## mu[5]       1
## mu[6]       1
## sigma[1]    1
## sigma[2]    1
## sigma[3]    1
## sigma[4]    1
## sigma[5]    1
## sigma[6]    1
## ypred[1]    1
## ypred[2]    1
## ypred[3]    1
## ypred[4]    1
## ypred[5]    1
## ypred[6]    1
## lp__        1
##
## Samples were drawn using NUTS(diag_e) at Sun Nov 03 22:14:07 2019.
```

```
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
separate_draws <- as.data.frame(extract(separate_result, pars = c("mu[6]", "ypred[6]"), permuted = T))

hist_post_mean <- ggplot(separate_draws, aes(mu.6.)) + geom_histogram(bins=20)
hist_post_mean
```




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



When modeling machines separately, it is not possible to plot a histogram for a seventh machine. The variance of the future observation is larger than the histogram of the mean, as expected.

POOLED MODEL

```
# STAN CODE
writeLines(readLines("model_pooled.stan"))
```

```
## data {
##   int<lower=0> N;
##   vector[N] y;
## }
## parameters {
##   real mu;
##   real<lower=0> sigma;
## }
## model {
##   y ~ normal(mu, sigma);
## }
## generated quantities {
##   real ypred;
##   ypred = normal_rng(mu, sigma);
## }
```

```
N <- 30
y <- c(factory$V1, factory$V2, factory$V3,
```

```
      factory$V4, factory$V5, factory$V6)
data_pooled <- list(N = N, y = y)

pooled_result <- stan("model_pooled.stan", data=data_pooled)
```

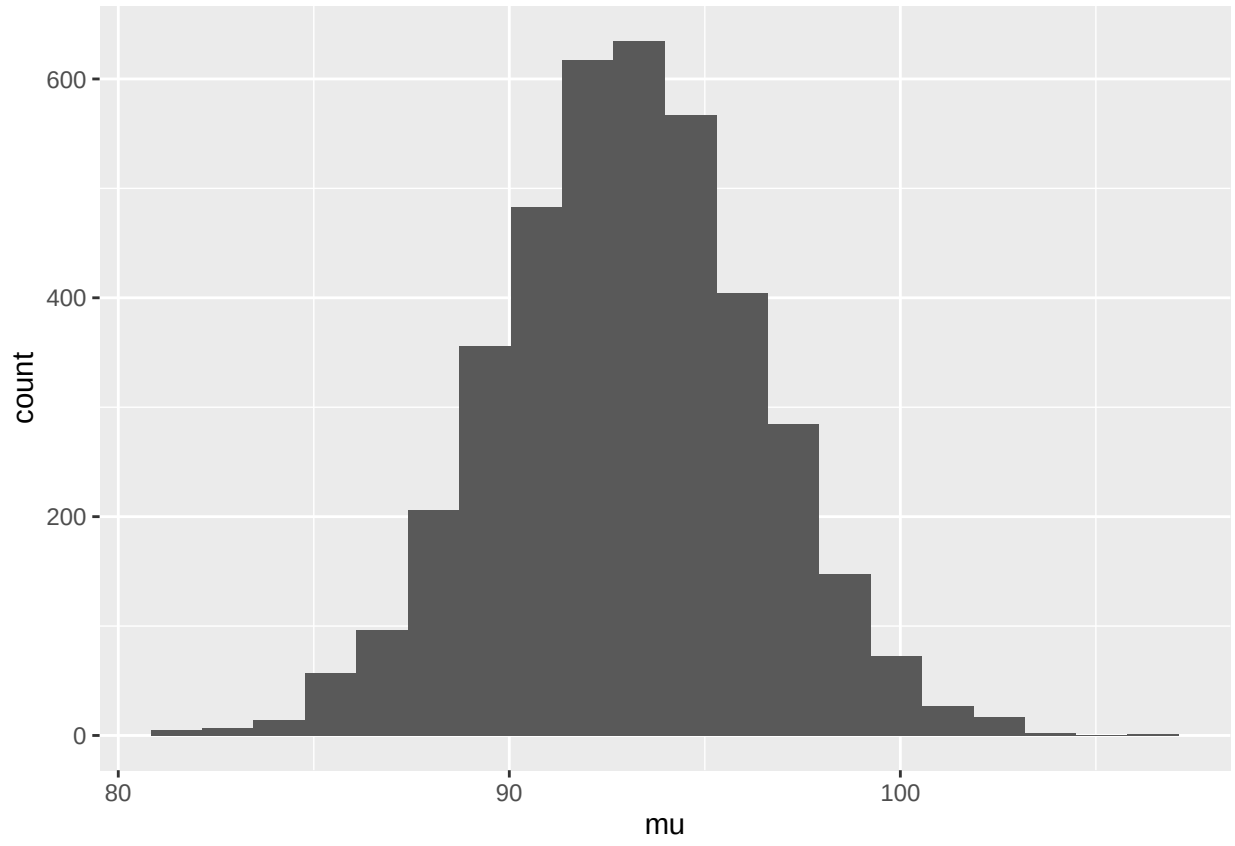
```
##
## SAMPLING FOR MODEL 'model_pooled' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    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: 0.033 seconds (Warm-up)
## Chain 1:                0.026 seconds (Sampling)
## Chain 1:                0.059 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'model_pooled' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.034 seconds (Warm-up)
## Chain 2:                0.019 seconds (Sampling)
```

```
## Chain 2:                0.053 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'model_pooled' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.034 seconds (Warm-up)
## Chain 3:                0.021 seconds (Sampling)
## Chain 3:                0.055 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'model_pooled' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.038 seconds (Warm-up)
## Chain 4:                0.029 seconds (Sampling)
## Chain 4:                0.067 seconds (Total)
## Chain 4:
```

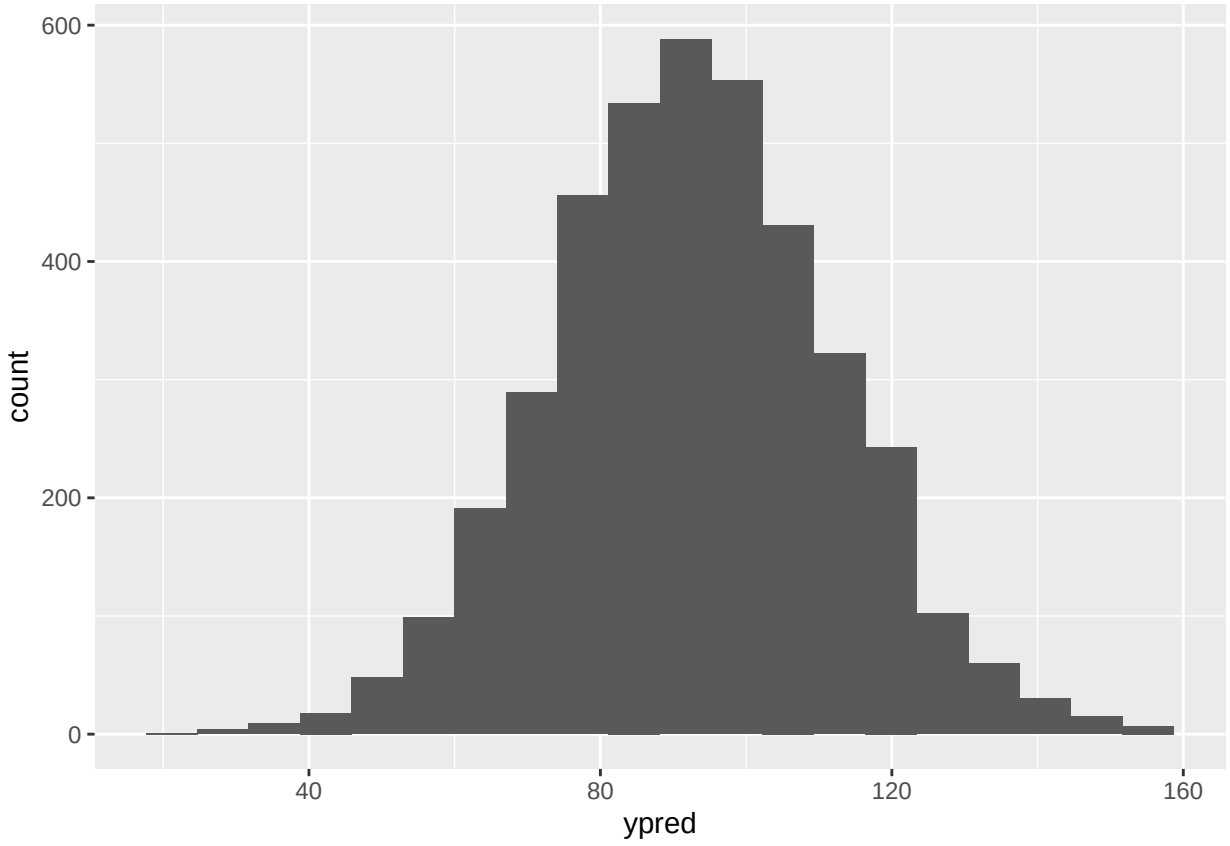
```
show(pooled_result)
```

```
## Inference for Stan model: model_pooled.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##         mean se_mean    sd    2.5%    25%    50%    75%  97.5% n_eff Rhat
## mu     92.99    0.06  3.31   86.47  90.79  92.98  95.17  99.51  2995    1
## sigma  18.73    0.05  2.50   14.54  16.98  18.51  20.20  24.27  2816    1
## ypred  92.65    0.31 19.49   54.76  79.49  92.45 105.89 131.06  3974    1
## lp__  -99.28    0.02  0.95 -101.83 -99.64 -98.99 -98.60 -98.35  1685    1
##
## Samples were drawn using NUTS(diag_e) at Sun Nov 03 22:15:22 2019.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
pooled_draws <- as.data.frame(extract(pooled_result, pars = c("mu", "ypred"), permuted = T))

hist_post_mean <- ggplot(pooled_draws, aes(mu)) + geom_histogram(bins=20)
hist_post_mean
```




```
hist_pred_y <- ggplot(pooled_draws, aes(ypred)) + geom_histogram(bins=20)
hist_pred_y
```



Here the posterior distribution of the mean of the quality measurements of the seventh machine is actually the same as the sixth machines. As we model this if all the observations come from same distribution, we assume that all machines follow the histogram that is illustrated in the first plot.

HIERARCHIAL MODEL

```
# STAN CODE
writeLines(readLines("model_hier.stan"))
```

```
## data {
##     int<lower=0> N; // number of data points
##     int<lower=0> K; // number of groups
##     int<lower=1,upper=K> x[N]; // group indicator
##     vector[N] y; //
## }
## parameters {
##     real mu0;             // prior mean
##     real<lower=0> sigma0; // prior std
##     vector[K] mu;         // group means
##     real<lower=0> sigma;  // common std
## }
## model {
##     mu0 ~ normal(95,10);      // weakly informative prior
##     sigma0 ~ cauchy(0,4);     // weakly informative prior
##     mu ~ normal(mu0, sigma0); // population prior with unknown parameters
##     sigma ~ cauchy(0,4);      // weakly informative prior
```

```
##     y ~ normal(mu[x], sigma);
## }
## generated quantities {
##   vector[K] ypred;
##   real mu7;
##
##   for (i in 1:K)
##     ypred[i] = normal_rng(mu[i], sigma);
##
##   mu7 = normal_rng(mu0, sigma0);
## }
```

```
N <- 30
K <- 6
x <- rep(1:ncol(factory), nrow(factory))
y <- y <- c(factory$V1, factory$V2, factory$V3,
            factory$V4, factory$V5, factory$V6)
data_hier <- list(N = N, K = K, x=x, y = y)

hier_result <- stan("model_hier.stan", data=data_hier)
```

```
##
## SAMPLING FOR MODEL 'model_hier' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.205 seconds (Warm-up)
## Chain 1:                0.201 seconds (Sampling)
## Chain 1:                0.406 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'model_hier' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.187 seconds (Warm-up)
## Chain 2:                0.167 seconds (Sampling)
## Chain 2:                0.354 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'model_hier' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.207 seconds (Warm-up)
## Chain 3:                0.144 seconds (Sampling)
## Chain 3:                0.351 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'model_hier' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.147 seconds (Warm-up)
## Chain 4:                0.143 seconds (Sampling)
## Chain 4:                0.29 seconds (Total)
## Chain 4:
```

```
## Warning: There were 199 divergent transitions after warmup. Increasing adapt_delta above 0.8 may help
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

```
## Warning: There were 1 chains where the estimated Bayesian Fraction of Missing Information was low. Se
## http://mc-stan.org/misc/warnings.html#bfmi-low
```

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

```
## Warning: The largest R-hat is 1.1, indicating chains have not mixed.
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#r-hat
```

```
## 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
```

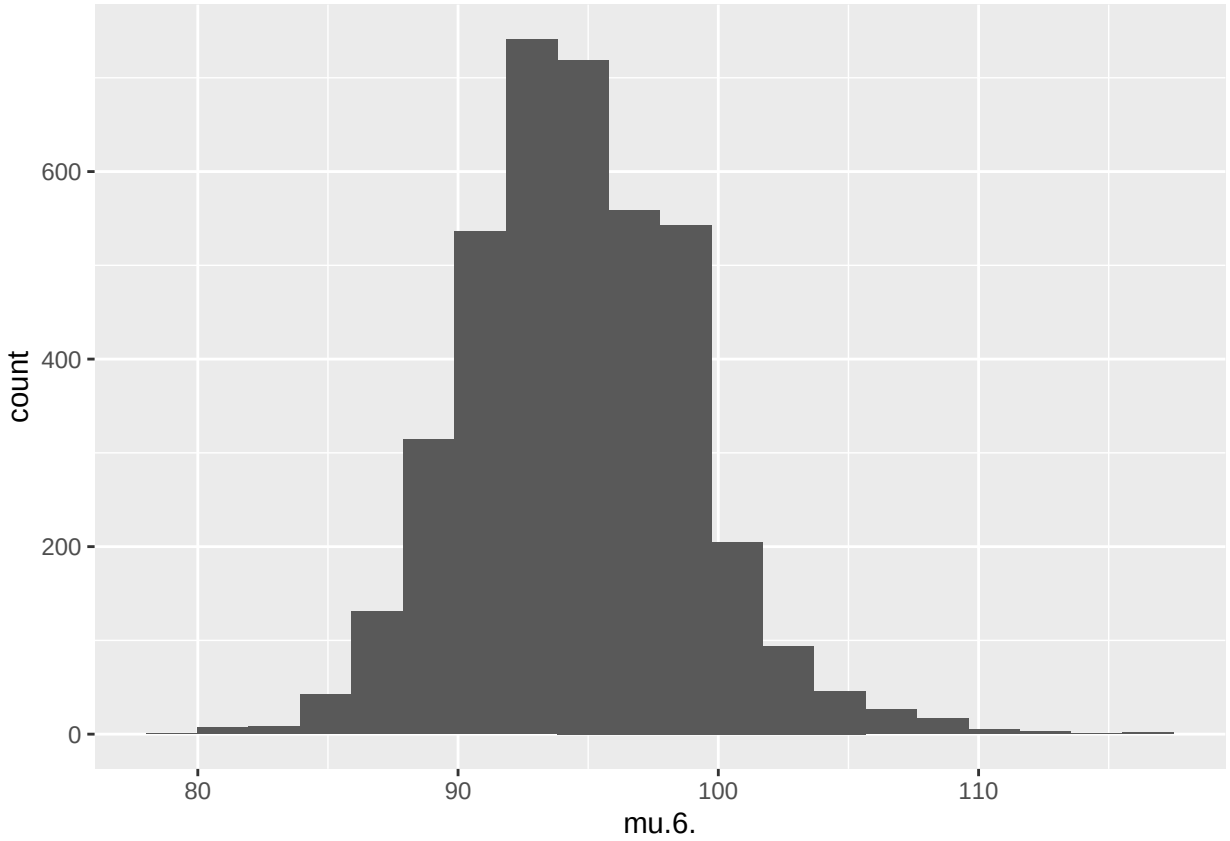
```
## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quanti
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#tail-ess
```

```
show(hier_result)
```

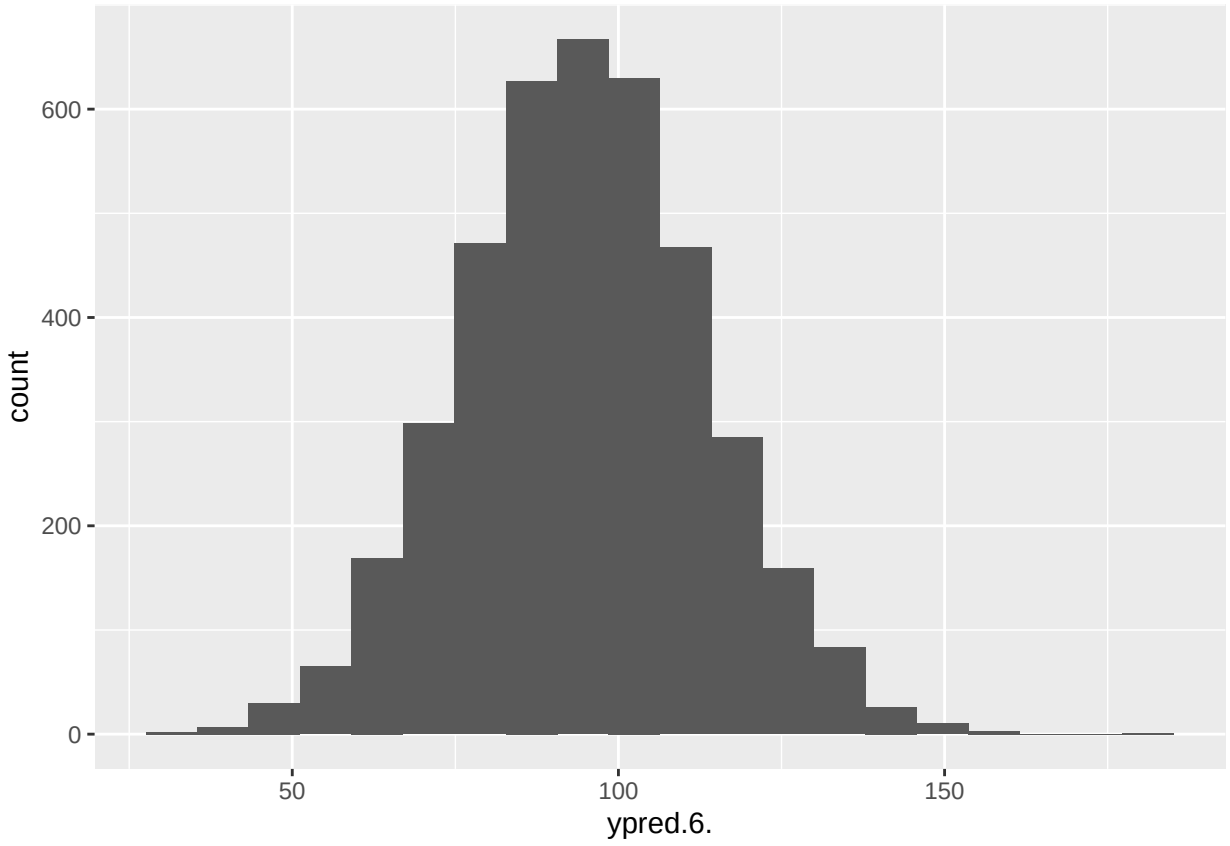
```
## Inference for Stan model: model_hier.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##             mean se_mean    sd    2.5%     25%     50%     75%  97.5%
## mu0        93.45    0.31  3.52   86.99   90.96   93.32   95.88  99.80
## sigma0      3.01    0.19  2.47    0.24    1.25    2.39    4.06   9.32
## mu[1]      93.83    0.27  4.29   85.89   90.94   93.72   96.82 102.26
## mu[2]      93.44    0.28  4.32   84.65   90.62   93.35   96.43 101.57
## mu[3]      92.50    0.39  4.61   82.50   89.70   92.58   95.57 100.33
## mu[4]      92.62    0.38  4.56   82.86   89.92   92.69   95.74 100.35
## mu[5]      93.51    0.30  4.21   85.40   90.71   93.39   96.48 101.49
## mu[6]      94.62    0.20  4.31   86.84   91.68   94.38   97.56 103.66
## sigma      18.20    0.07  2.42   14.22   16.45   17.90   19.64  23.80
## ypred[1]   93.41    0.30 18.78   56.63   80.98   93.66  105.82 131.09
## ypred[2]   93.59    0.30 18.74   56.84   81.04   93.84  105.92 130.87
```

```
## ypred[3]   92.89    0.32 19.08   54.85   80.46   92.88  105.50 131.18
## ypred[4]   92.20    0.36 18.84   54.78   80.04   91.90  104.93 129.04
## ypred[5]   93.68    0.30 18.63   55.94   81.58   93.67  106.23 129.42
## ypred[6]   94.56    0.31 18.47   58.61   82.13   94.18  106.85 131.40
## mu7        93.45    0.26  5.17   82.81   90.40   93.47   96.82 103.53
## lp__     -109.55    0.85  5.18 -119.12 -113.10 -109.84 -106.26 -98.40
##          n_eff Rhat
## mu0        128 1.04
## sigma0     165 1.04
## mu[1]      243 1.02
## mu[2]      233 1.03
## mu[3]      141 1.03
## mu[4]      148 1.03
## mu[5]      193 1.03
## mu[6]      473 1.02
## sigma     1050 1.01
## ypred[1]  3849 1.00
## ypred[2]  3986 1.00
## ypred[3]  3509 1.00
## ypred[4]  2690 1.00
## ypred[5]  3880 1.00
## ypred[6]  3609 1.00
## mu7        392 1.02
## lp__        38 1.11
##
## Samples were drawn using NUTS(diag_e) at Sun Nov 03 22:16:31 2019.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

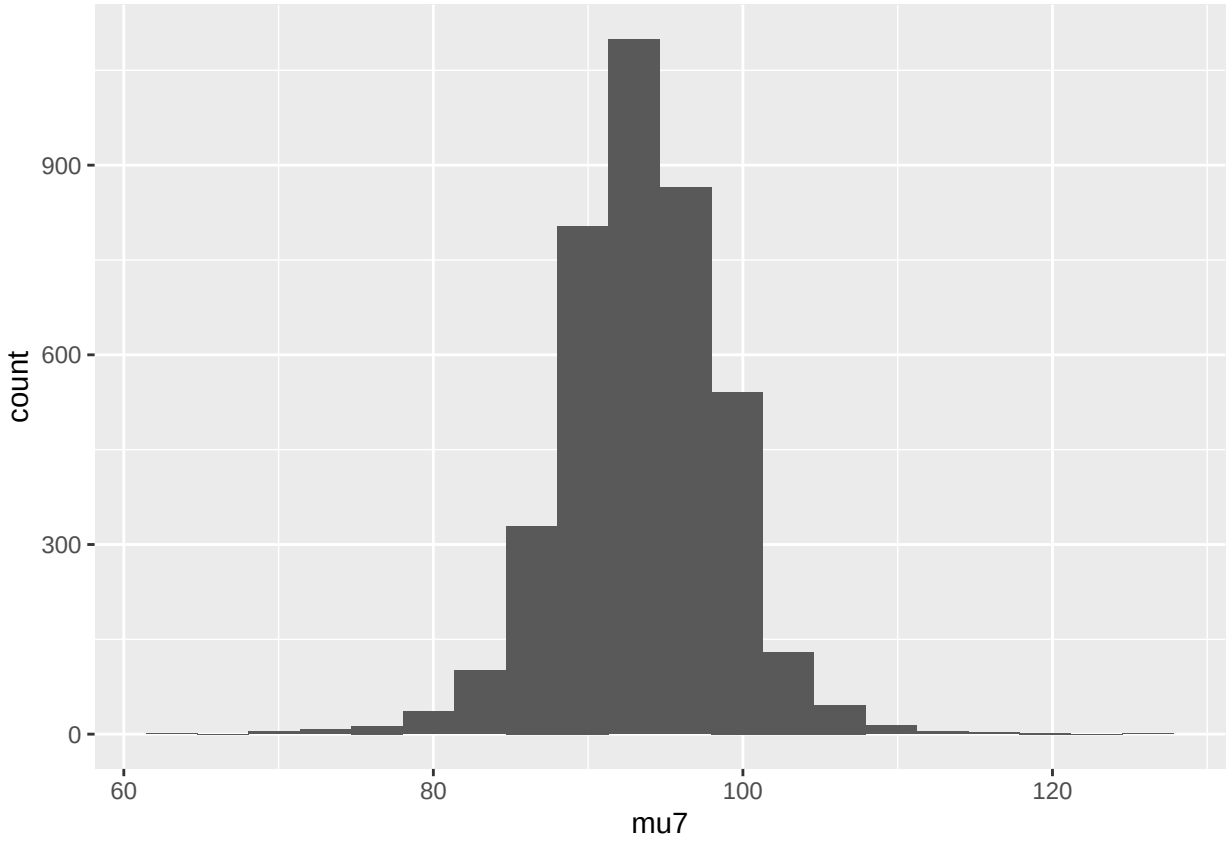
```
hier_draws <- as.data.frame(extract(hier_result, pars = c("mu[6]", "mu7", "ypred[6]"), permuted = T))

hist_post_mean <- ggplot(hier_draws, aes(mu.6.)) + geom_histogram(bins=20)
hist_post_mean
```




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




```
hist_post_mean_seven <- ggplot(hier_draws, aes(mu7)) + geom_histogram(bins=20)
hist_post_mean_seven
```



Do not let the scale mislead you, the posterior distribution of the mean for seventh machine is wider. The future observation for machine six follows a normal distribution illustrated in the second plot. The interval is, in fact, very large.