BDA - Assignment 7

7/10/2021

1

a)

The following things have been fixed 1. The bound of sigma, as the variance can't be negative we added a lowr bound of 0 to it. 2. We added a ; to the row for y in the model block for the model to run. 3. We changed x to xpred for ypred in the generated quantitites. (4. added and empty row at the end for the stan file to run)

Fixed are marked with comments below.

```
data {
  int<lower=0> N;
  vector[N] x;
  vector[N] y;
  real xpred;
}
parameters {
  real alpha;                // added at part d)
  real beta;                 // added at part c)
  real<lower=0> sigma;       // fixed at part a)
  real<lower=0> sigma_beta; // added at part c)
  real mu_alpha;
  real<lower=0> sigma_alpha; // added at part d)
}
transformed parameters {
  vector[N] mu = alpha + beta*x;
}
model {
  alpha ~ normal(mu_alpha,sigma_alpha); // added at part d)
  beta ~ normal(0, sigma_beta); // added at part c)
  y ~ normal(mu, sigma); // fixed at part a)
}
generated quantities {
  real ypred = normal_rng(alpha + beta*xpred, sigma); // fixed at part a)
}
```

b)

We determine a weekly informative beta by using the normal scaling factor z. In order to get the 99%-interval we use qnorm of 0.995.

```
mu_hist = 138
mu_beta = 0
z = qnorm(0.995)
sigma_beta = (mu_hist/2-mu_beta)/z
```

c)

The parameters sigma_beta was added as well as beta to the (parameter and) model block. The additions are marked in the model with the comment "// added at part c)"

d)

```
z = qnorm(0.995)
test=lm(formula = drowning$drownings ~ drowning$year)
mu_alpha = lm(formula = drowning$drownings ~ drowning$year)$coefficients[[1]]
sigma_alpha = (mu_alpha/2 - 0)/z
```

We use a linear fit and similar technique as in c part to create a weekly prior alpha. The additions in the coded are marked with "// added at part d)"

```
mu <- mean(drowning[,2])
sigma <- var(drowning[,2])
data <- list(
  N = length(drowning$drownings), #Number of data points
  x = drowning$year,              #Year
  y = drowning$drownings,         #Amount of drownings
  xpred = 2020,                   #Year(s) to predict
  #mu = mu,                        #Mean vector
  #sigma = sigma,                  #Covariance matrix
  simga_beta=sigma_beta,          #Variance of beta
  mu_alpha = mu_alpha,
  sigma_alpha = sigma_alpha
)
```

```
fit1 = sampling(stanmodel,
  data = data,            # named list of data
  chains = 4,             # number of Markov chains
  warmup = 1000,          # number of warmup iterations per chain
  iter = 2000,            # total number of iterations per chain
  cores = 1,              # number of cores (could use one per chain)
  refresh = 0             # no progress shown
  )
```

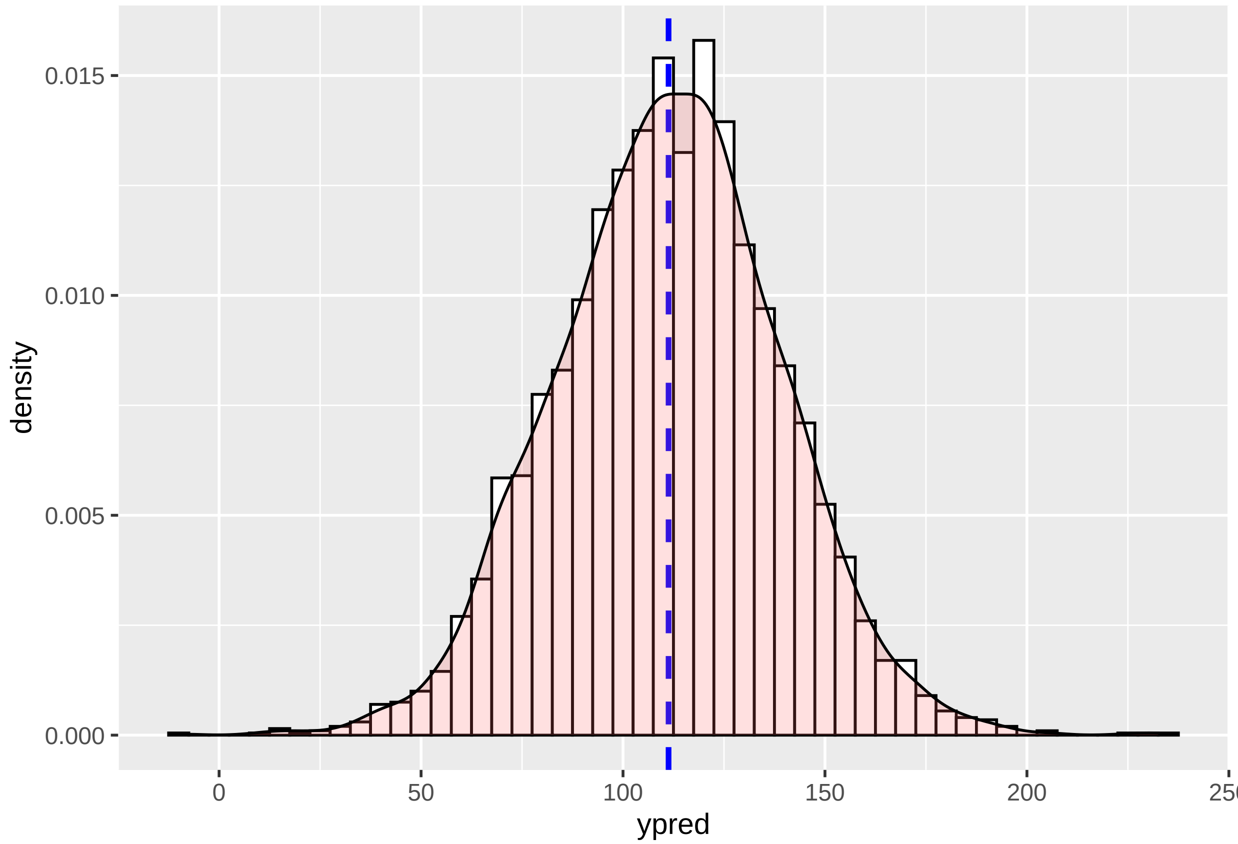
```
print(fit1)
```

```
## Inference for Stan model: 834b141f3cd4e2c48f16745cbca435f0.
## 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%
## alpha        2.434160e+03      25.97     705.63  1.019860e+03  1.976170e+03
## beta        -1.150000e+00       0.01       0.35 -1.840000e+00 -1.390000e+00
## sigma        2.623000e+01       0.10       3.24  2.088000e+01  2.397000e+01
## sigma_beta   1.765719e+304       NaN        Inf  1.886196e+07  5.839288e+70
## mu_alpha     1.945420e+06 1571448.29 2863648.69 -2.832120e+06  6.061125e+05
## sigma_alpha 1.497788e+305       NaN        Inf  7.381632e+13  4.128619e+79
## mu[1]        1.567400e+02       0.27       8.04  1.406600e+02  1.514900e+02
## mu[2]        1.555900e+02       0.25       7.74  1.400700e+02  1.505400e+02
## mu[3]        1.544400e+02       0.24       7.44  1.394600e+02  1.495900e+02
## mu[4]        1.532900e+02       0.23       7.15  1.388500e+02  1.486200e+02
## mu[5]        1.521400e+02       0.22       6.86  1.381400e+02  1.476500e+02
```

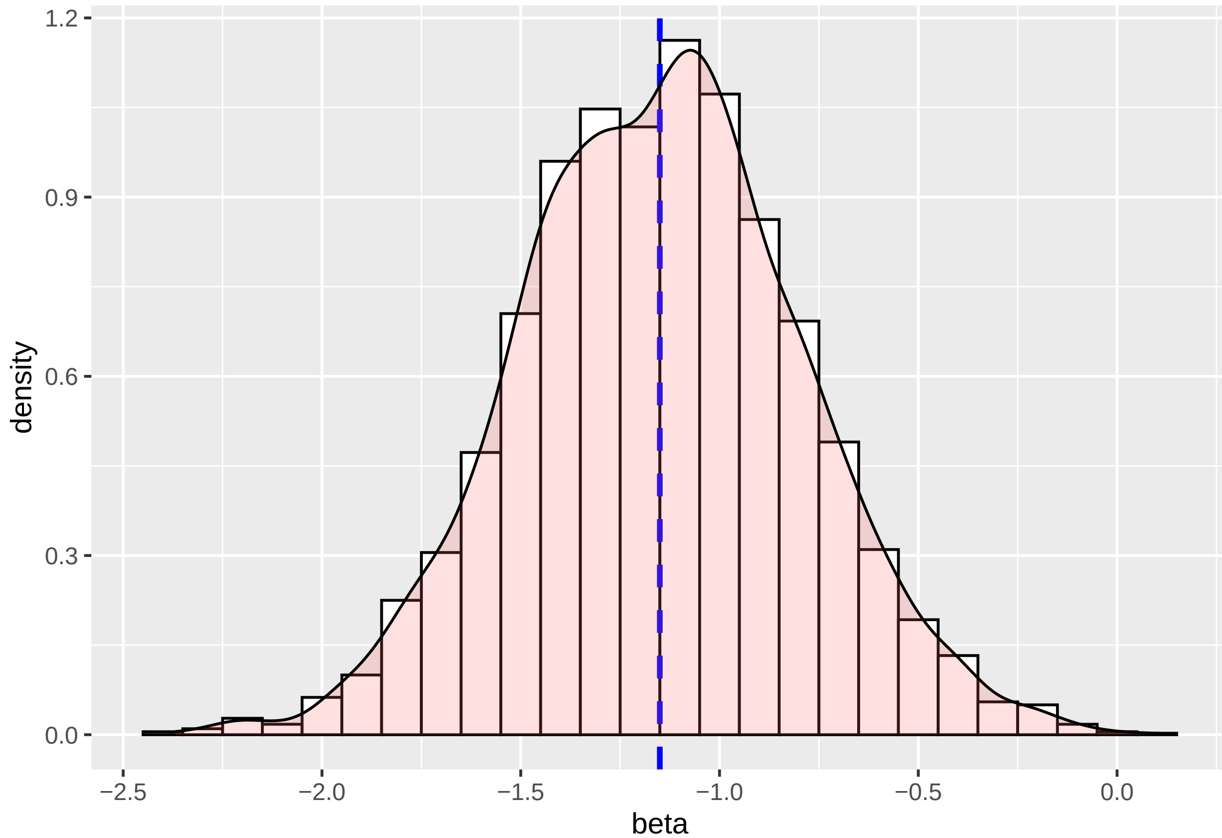
```
## mu[6]          1.509900e+02       0.20       6.58  1.375000e+02  1.466500e+02
## mu[7]          1.498400e+02       0.19       6.31  1.368600e+02  1.456800e+02
## mu[8]          1.486900e+02       0.18       6.05  1.363400e+02  1.446700e+02
## mu[9]          1.475400e+02       0.17       5.79  1.357400e+02  1.437200e+02
## mu[10]         1.463900e+02       0.15       5.55  1.351900e+02  1.427400e+02
## mu[11]         1.452400e+02       0.14       5.32  1.344900e+02  1.417300e+02
## mu[12]         1.440900e+02       0.13       5.10  1.338800e+02  1.407400e+02
## mu[13]         1.429400e+02       0.12       4.90  1.331600e+02  1.397200e+02
## mu[14]         1.417900e+02       0.11       4.72  1.323900e+02  1.386800e+02
## mu[15]         1.406400e+02       0.10       4.56  1.315900e+02  1.375900e+02
## mu[16]         1.394900e+02       0.09       4.42  1.307700e+02  1.365500e+02
## mu[17]         1.383400e+02       0.08       4.30  1.298400e+02  1.355300e+02
## mu[18]         1.371900e+02       0.07       4.21  1.288400e+02  1.344700e+02
## mu[19]         1.360400e+02       0.07       4.15  1.279100e+02  1.333700e+02
## mu[20]         1.348900e+02       0.07       4.12  1.268400e+02  1.322300e+02
## mu[21]         1.337400e+02       0.06       4.12  1.256100e+02  1.310400e+02
## mu[22]         1.325900e+02       0.07       4.14  1.242600e+02  1.298600e+02
## mu[23]         1.314400e+02       0.07       4.20  1.230300e+02  1.286700e+02
## mu[24]         1.302900e+02       0.07       4.29  1.217300e+02  1.274600e+02
## mu[25]         1.291400e+02       0.08       4.40  1.203900e+02  1.262400e+02
## mu[26]         1.279900e+02       0.09       4.54  1.190100e+02  1.250500e+02
## mu[27]         1.268400e+02       0.10       4.70  1.175800e+02  1.237400e+02
## mu[28]         1.256900e+02       0.11       4.88  1.160500e+02  1.224800e+02
## mu[29]         1.245400e+02       0.12       5.08  1.145100e+02  1.212000e+02
## mu[30]         1.233900e+02       0.13       5.29  1.128400e+02  1.198900e+02
## mu[31]         1.222400e+02       0.14       5.52  1.112900e+02  1.185700e+02
## mu[32]         1.210900e+02       0.15       5.76  1.096800e+02  1.172300e+02
## mu[33]         1.199400e+02       0.17       6.02  1.080400e+02  1.159600e+02
## mu[34]         1.187900e+02       0.18       6.28  1.063300e+02  1.146600e+02
## mu[35]         1.176400e+02       0.19       6.55  1.047100e+02  1.133200e+02
## mu[36]         1.164900e+02       0.20       6.83  1.030900e+02  1.119700e+02
## mu[37]         1.153400e+02       0.22       7.11  1.013400e+02  1.105800e+02
## mu[38]         1.141800e+02       0.23       7.40  9.963000e+01  1.092500e+02
## mu[39]         1.130300e+02       0.24       7.70  9.784000e+01  1.079300e+02
## mu[40]         1.118800e+02       0.25       8.00  9.605000e+01  1.065600e+02
## ypred          1.112800e+02       0.49      27.65  5.782000e+01  9.330000e+01
## lp__          -1.466900e+02       0.05       1.29 -1.500900e+02 -1.472600e+02
##                          50%            75%          97.5% n_eff Rhat
## alpha          2.424480e+03   2.916170e+03   3.807260e+03   738 1.00
## beta          -1.150000e+00  -9.200000e-01  -4.400000e-01   739 1.00
## sigma          2.585000e+01   2.811000e+01   3.380000e+01  1033 1.01
## sigma_beta    1.198564e+147  3.287361e+222  3.307522e+293   NaN  NaN
## mu_alpha       1.505886e+06   2.707897e+06   1.010993e+07     3 2.20
## sigma_alpha   2.465545e+151  3.111098e+224  7.758369e+300   NaN  NaN
## mu[1]          1.568000e+02   1.621500e+02   1.723100e+02   907 1.00
## mu[2]          1.556800e+02   1.607700e+02   1.704500e+02   926 1.00
## mu[3]          1.545200e+02   1.594200e+02   1.687700e+02   948 1.00
## mu[4]          1.533900e+02   1.580700e+02   1.669500e+02   974 1.00
## mu[5]          1.522400e+02   1.567500e+02   1.652100e+02  1005 1.00
## mu[6]          1.510700e+02   1.554400e+02   1.636500e+02  1041 1.00
## mu[7]          1.499500e+02   1.541000e+02   1.619100e+02  1086 1.00
## mu[8]          1.487700e+02   1.527800e+02   1.602400e+02  1141 1.00
## mu[9]          1.476400e+02   1.514200e+02   1.587200e+02  1209 1.00
## mu[10]         1.465300e+02   1.500900e+02   1.570900e+02  1292 1.00
```

```
## mu[11]        1.453700e+02   1.488300e+02   1.553400e+02  1398 1.00
## mu[12]        1.441800e+02   1.475800e+02   1.537900e+02  1533 1.00
## mu[13]        1.430100e+02   1.462600e+02   1.523200e+02  1710 1.00
## mu[14]        1.418400e+02   1.449500e+02   1.508100e+02  1933 1.00
## mu[15]        1.406300e+02   1.437100e+02   1.493600e+02  2203 1.00
## mu[16]        1.394600e+02   1.424500e+02   1.479400e+02  2527 1.00
## mu[17]        1.383500e+02   1.411900e+02   1.465800e+02  2947 1.00
## mu[18]        1.371800e+02   1.399600e+02   1.453600e+02  3330 1.00
## mu[19]        1.360200e+02   1.387600e+02   1.441700e+02  3690 1.00
## mu[20]        1.348900e+02   1.375800e+02   1.431900e+02  3964 1.00
## mu[21]        1.337300e+02   1.364400e+02   1.421200e+02  4029 1.00
## mu[22]        1.325500e+02   1.352600e+02   1.410500e+02  3934 1.00
## mu[23]        1.313900e+02   1.341200e+02   1.399100e+02  3714 1.00
## mu[24]        1.302500e+02   1.330600e+02   1.389100e+02  3393 1.00
## mu[25]        1.290700e+02   1.320300e+02   1.380800e+02  3036 1.00
## mu[26]        1.279200e+02   1.309700e+02   1.372100e+02  2707 1.00
## mu[27]        1.267700e+02   1.299400e+02   1.363700e+02  2304 1.00
## mu[28]        1.256200e+02   1.289400e+02   1.354900e+02  2022 1.00
## mu[29]        1.244700e+02   1.279500e+02   1.347100e+02  1808 1.00
## mu[30]        1.233200e+02   1.269600e+02   1.339500e+02  1639 1.00
## mu[31]        1.221600e+02   1.260400e+02   1.332300e+02  1506 1.00
## mu[32]        1.210400e+02   1.250200e+02   1.326300e+02  1399 1.00
## mu[33]        1.199200e+02   1.240500e+02   1.318900e+02  1313 1.00
## mu[34]        1.187900e+02   1.230000e+02   1.312100e+02  1242 1.00
## mu[35]        1.176500e+02   1.219800e+02   1.305800e+02  1184 1.00
## mu[36]        1.165200e+02   1.210200e+02   1.298400e+02  1134 1.00
## mu[37]        1.154300e+02   1.200300e+02   1.293400e+02  1092 1.00
## mu[38]        1.143100e+02   1.191400e+02   1.286900e+02  1056 1.00
## mu[39]        1.131800e+02   1.182000e+02   1.280400e+02  1025 1.00
## mu[40]        1.120400e+02   1.172600e+02   1.274900e+02   999 1.00
## ypred         1.116700e+02   1.293600e+02   1.647300e+02  3246 1.00
## lp__         -1.463300e+02  -1.457600e+02  -1.452600e+02   741 1.00
##
## Samples were drawn using NUTS(diag_e) at Fri Nov 12 12:58:26 2021.
## 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).
```

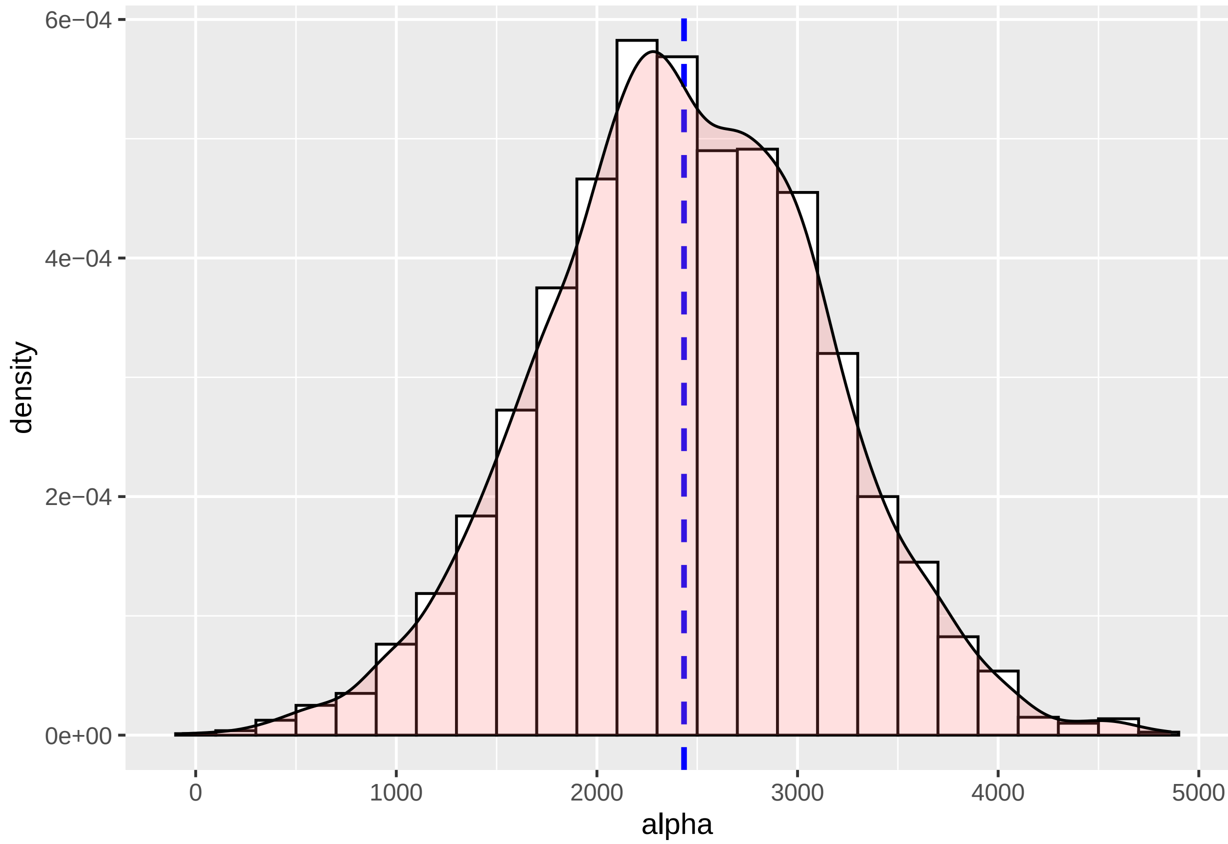
```r
extract <- data.frame(extract(fit1))
ggplot(extract, aes(x=ypred)) +
  geom_histogram(aes(y=..density..), binwidth = 5, colour="black", fill="white") +
  geom_vline(aes(xintercept=mean(ypred)), color="blue", linetype="dashed", size=1) +
  geom_density(alpha=.2, fill="#FF6666")
```




```
ggplot(extract, aes(x=beta)) +
  geom_histogram(aes(y=..density..), binwidth = 0.1, colour="black", fill="white") +
  geom_vline(aes(xintercept=mean(beta)), color="blue", linetype="dashed", size=1) +
  geom_density(alpha=.2, fill="#FF6666")
```




```
ggplot(extract, aes(x=alpha)) +   geom_histogram(aes(y=..density..), binwidth = 200, colour="black", fil
  geom_vline(aes(xintercept=mean(alpha)), color="blue", linetype="dashed", size=1) +
  geom_density(alpha=.2, fill="#FF6666")
```



The histograms looks similar to those in the assignment.

2

```
rm(list = ls())
data("factory")
```

Separate model

a)

In the separate model, we assume that all the machines come from their own indepdendent distribution. The mathematical formulation is

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

$\mu_j \sim \mathcal{N}(100, 10)$

$\sigma_j \sim \text{Inv-}\chi^2(10).$

b)

```
data {
 int<lower=0> N;
 int<lower=0> J;
 vector[J] y[N];
 real mean_mu_prior;
```

```
 real<lower=0> mean_sigma_prior;
 real<lower=0> sigma_prior;
}

parameters {
 vector[J] mu;
 vector <lower=0>[J] sigma;
}

model {
 //priors
 for (j in 1:J){
  mu[j] ~ normal(mean_mu_prior, mean_sigma_prior);
  sigma[j] ~ inv_chi_square(sigma_prior);
 }
 //likelihoods
 for (j in 1:J)
  y[,j] ~ normal(mu[j], sigma[j]);
 }

generated quantities {
 //for the first machine
 real ypred;
 ypred = normal_rng(mu[6], sigma[6]);
}
```

```r
mean_mu_prior = 100
mean_sigma_prior = 10
sigma_prior = 10
separate_data <- list(
  y = factory,
  N = nrow(factory),
  J = ncol(factory),
 mean_mu_prior = mean_mu_prior,
 mean_sigma_prior = mean_sigma_prior,
 sigma_prior = sigma_prior
)
```

```r
fit_separate = sampling(separatemodel,
  data = separate_data,            # named list of data
  chains = 4,             # number of Markov chains
  warmup = 1000,          # number of warmup iterations per chain
  iter = 2000,            # total number of iterations per chain
  cores = 1,              # number of cores (could use one per chain)
  refresh = 0             # no progress shown
  )
```

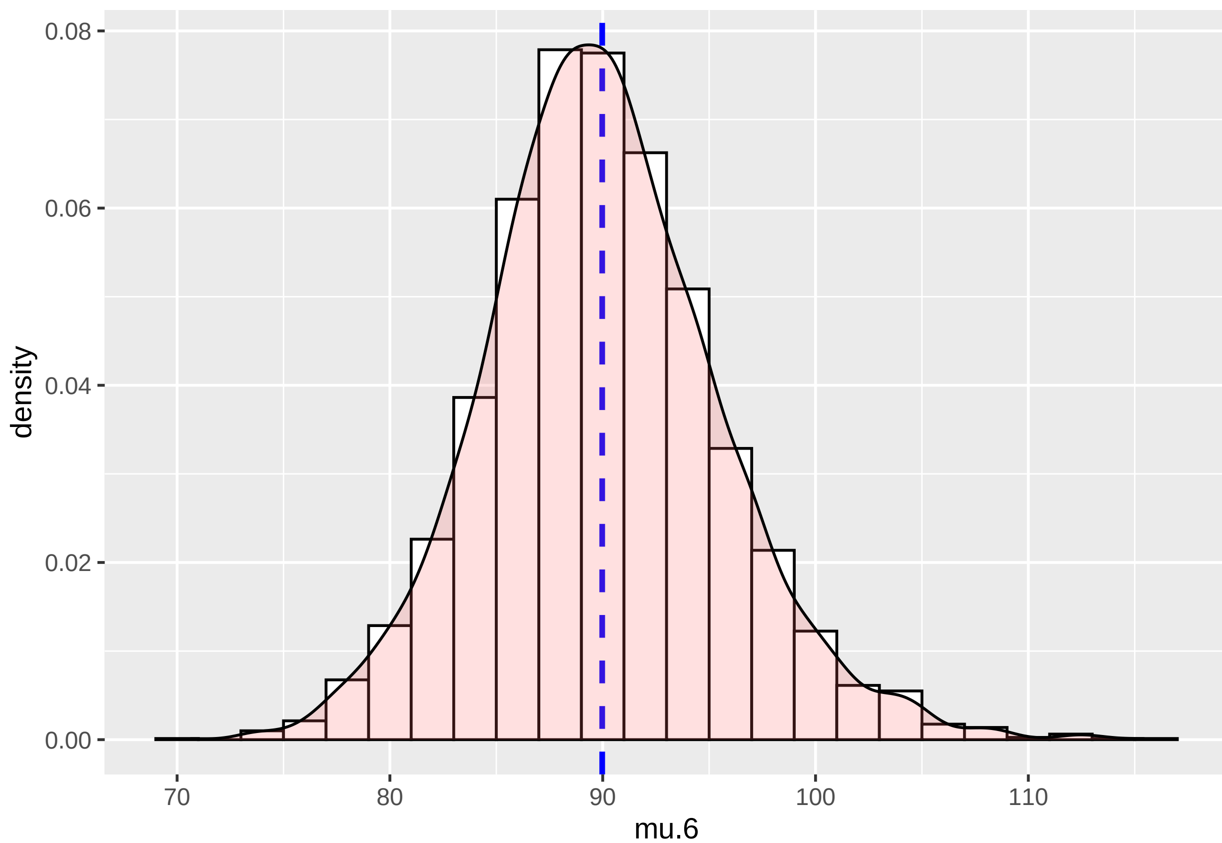
```r
print(fit_separate)
```

```
## Inference for Stan model: 9ab7f66dd32347bb768dbdb565f8f979.
## 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]       83.49    0.10 6.30   72.30   79.15   83.09   87.36   97.01  3915
```

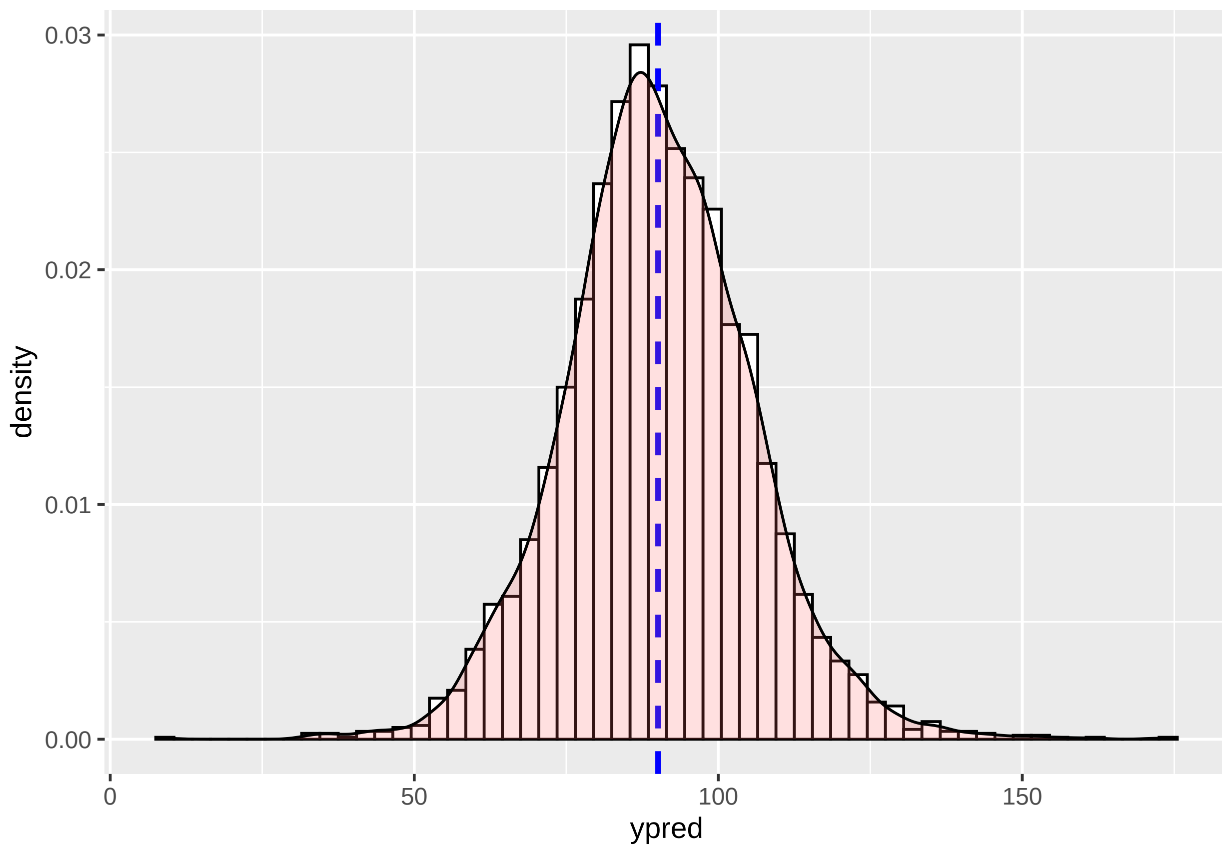
```
## mu[2]      105.35    0.05  3.62   98.04  103.14  105.41  107.67  112.27  4709
## mu[3]       89.68    0.06  3.96   82.43   87.05   89.47   92.03   98.20  4416
## mu[4]      110.89    0.04  2.48  105.93  109.33  110.93  112.47  115.68  4309
## mu[5]       91.20    0.06  3.53   84.68   88.96   91.07   93.29   98.49  3966
## mu[6]       89.98    0.08  5.52   79.43   86.41   89.74   93.32  101.70  4748
## sigma[1]    15.36    0.08  4.62    9.35   12.14   14.35   17.66   26.40  3276
## sigma[2]     8.62    0.04  2.32    5.39    7.01    8.19    9.75   14.30  3677
## sigma[3]     9.29    0.04  2.46    5.91    7.55    8.85   10.46   15.67  3746
## sigma[4]     5.57    0.03  1.57    3.45    4.50    5.30    6.29    9.28  3407
## sigma[5]     7.91    0.04  2.16    4.93    6.40    7.51    8.98   13.09  3333
## sigma[6]    14.13    0.06  3.85    8.84   11.48   13.39   16.04   23.71  3676
## ypred       90.11    0.25 15.39   60.17   80.56   89.58   99.60  121.54  3847
## lp__      -167.82    0.07  2.66 -174.22 -169.40 -167.40 -165.85 -163.77  1449
##           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
## lp__         1
##
## Samples were drawn using NUTS(diag_e) at Fri Nov 12 12:58:54 2021.
## 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).
```

c)

```
extract_separate <- data.frame(extract(fit_separate))

ggplot(extract_separate, aes(x=mu.6)) +
  geom_histogram(aes(y=..density..), binwidth = 2, colour="black", fill="white") +
  geom_vline(aes(xintercept=mean(mu.6)), color="blue", linetype="dashed", size=1) +
  geom_density(alpha=.2, fill="#FF6666")
```




```
ggplot(extract_separate, aes(x=ypred)) +
  geom_histogram(aes(y=..density..), binwidth = 3, colour="black", fill="white") +
  geom_vline(aes(xintercept=mean(ypred)), color="blue", linetype="dashed", size=1) +
  geom_density(alpha=.2, fill="#FF6666")
```



i) We can see the distribution of the mean of the 6th machine in the first histogram. The mean of the distribution lies around 90.

ii) We can see the distribution of the predictions of the 6th machine in the second histogram.

iii) Since we have separate models, there's no way for us to accurately predict a 7th machines values.

d)

We make a new model in order to simulate the alternative priors.

```
data {
 int<lower=0> N;
 int<lower=0> J;
 vector[J] y[N];
 real mean_mu_prior;
 real<lower=0> mean_sigma_prior;
 real<lower=0> sigma_prior;
}

parameters {
 vector[J] mu;
 vector <lower=0>[J] sigma;
}

model {
```

```
 //priors
 for (j in 1:J){
  mu[j] ~ normal(mean_mu_prior, mean_sigma_prior);
  sigma[j] ~ gamma(sigma_prior, sigma_prior);
 }
 //likelihoods
 for (j in 1:J)
  y[,j] ~ normal(mu[j], sigma[j]);
 }

generated quantities {
 //for the sixth machine
 real ypred;
 ypred = normal_rng(mu[6], sigma[6]);
}
```

```
separate_data_alternative <- list(
  y = factory,
  N = nrow(factory),
  J = ncol(factory),
  mean_mu_prior = 0,
  mean_sigma_prior = 10,
  sigma_prior = 1
)

fit_separate_alternative = sampling(separatemodel_alternative,
  data = separate_data_alternative,            # named list of data
  chains = 4,             # number of Markov chains
  warmup = 1000,          # number of warmup iterations per chain
  iter = 2000,            # total number of iterations per chain
  cores = 1,              # number of cores (could use one per chain)
  refresh = 0             # no progress shown
  )

extract_separate_alternative <- data.frame(extract(fit_separate_alternative))

mu.1_interval = quantile(extract_separate_alternative$mu.1, probs=c(0.05,0.95))
```

The posterior expectation for the the mean of the first machine is 50.31 with a 90% credible interval of 34.2539984, 63.8471895.

Pooled model

a)

In the pooled model, we assume that all the samples come from one distribution with the same parameters θ. Thus, we only have one μ and σ for all the samples.

$y\_i \sim \mathcal{N}(\mu, \sigma)$

$\mu \sim \mathcal{N}(100, 10)$

$\sigma \sim \text{Inv-}\chi^2(10)$.

Thus, instead of looping over the machines, as we did in the previous part, we now draw all the y's from the one and same distribution in our stan code.

b)

```
data {
 int<lower=0> N;
 int<lower=0> J;
 vector[N*J] y;
 real mean_mu_prior;
 real<lower=0> mean_sigma_prior;
 real<lower=0> sigma_prior;
}

parameters {
 real mu;
 real <lower=0> sigma;
}

model {
 //prior
  mu ~ normal(mean_mu_prior, mean_sigma_prior);
  sigma ~ inv_chi_square(sigma_prior);
 //likelihoods
  y ~ normal(mu, sigma);
}

generated quantities {
 real ypred;
 ypred = normal_rng(mu, sigma);
}
```

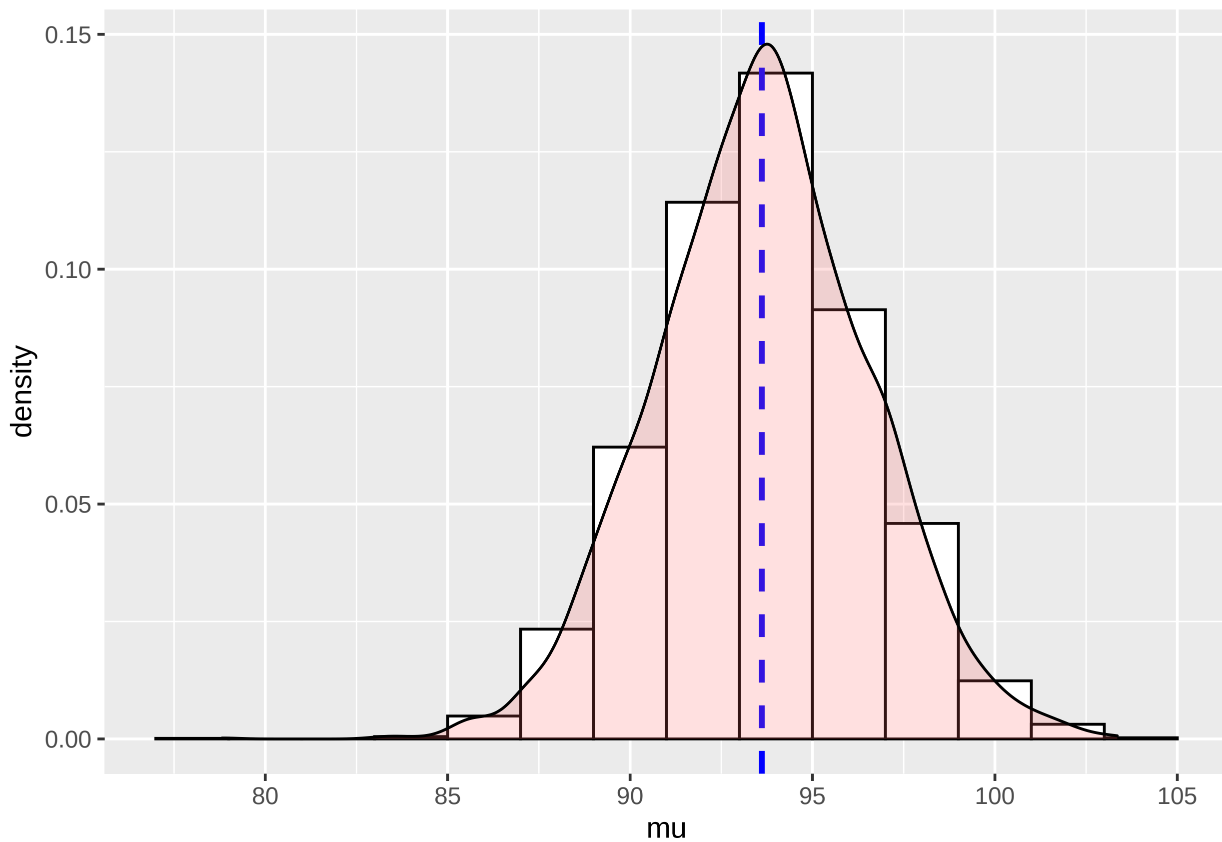
```r
mean_mu_prior = 100
mean_sigma_prior = 10
sigma_prior = 10
pooled_data <- list(
  y = unlist(factory),
  N = nrow(factory),
  J = ncol(factory),
 mean_mu_prior = mean_mu_prior,
 mean_sigma_prior = mean_sigma_prior,
 sigma_prior = sigma_prior
)
```

```r
fit_pooled = sampling(pooledmodel,
  data = pooled_data,            # named list of data
  chains = 4,             # number of Markov chains
  warmup = 1000,          # number of warmup iterations per chain
  iter = 2000,            # total number of iterations per chain
  cores = 1,              # number of cores (could use one per chain)
  refresh = 0             # no progress shown
  )
```

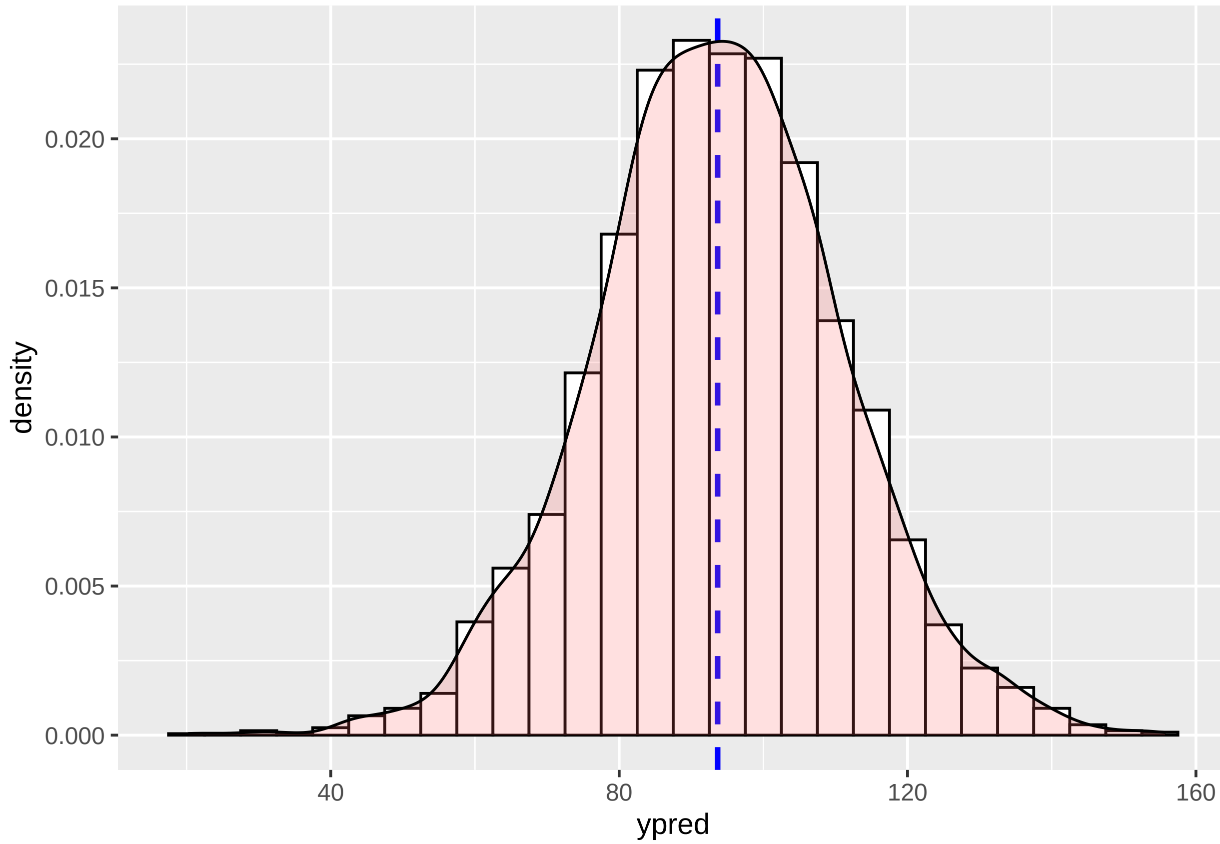
c)

```r
extract_pooled <- data.frame(extract(fit_pooled))

ggplot(extract_pooled, aes(x=mu)) +
```

```
  geom_histogram(aes(y=..density..), binwidth = 2, colour="black", fill="white") +
  geom_vline(aes(xintercept=mean(mu)), color="blue", linetype="dashed", size=1) +
  geom_density(alpha=.2, fill="#FF6666")
```




```
ggplot(extract_pooled, aes(x=ypred)) +
  geom_histogram(aes(y=..density..), binwidth = 5, colour="black", fill="white") +
  geom_vline(aes(xintercept=mean(ypred)), color="blue", linetype="dashed", size=1) +
  geom_density(alpha=.2, fill="#FF6666")
```



Since we have a pooled model, the posterior distribution for the machines are all the same.

i) We can see the distribution of the mean of the 6th machine in the first histogram. The mean of the distribution lies around 93.

ii) We can see the distribution of the predictions of the 6th machine in the second histogram.

iii) Since, as noted, the porsterioir distribution is the same ofr all the machines is the distribution of the mean for the seventh machine the same as the sixth machine.

d)

```
data {
 int<lower=0> N;
 int<lower=0> J;
 vector[N*J] y;
 real mean_mu_prior;
 real<lower=0> mean_sigma_prior;
 real<lower=0> sigma_prior;
}

parameters {
 real mu;
 real <lower=0> sigma;
}

model {
```

```
 //prior
  mu ~ normal(mean_mu_prior, mean_sigma_prior);
  sigma ~ gamma(sigma_prior,sigma_prior);
 //likelihoods
  y ~ normal(mu, sigma);
}

generated quantities {
 real ypred;
 ypred = normal_rng(mu, sigma);
}
```

```
pooled_data_alternative <- list(
  y = unlist(factory),
  N = nrow(factory),
  J = ncol(factory),
 mean_mu_prior = 1,
 mean_sigma_prior = 10,
 sigma_prior = 1
)

fit_pooled_alternative = sampling(pooledmodel_alternative,
  data = pooled_data_alternative,           # named list of data
  chains = 4,             # number of Markov chains
  warmup = 1000,          # number of warmup iterations per chain
  iter = 2000,            # total number of iterations per chain
  cores = 1,              # number of cores (could use one per chain)
  refresh = 0             # no progress shown
)

extract_pooled_alternative <- data.frame(extract(fit_pooled_alternative))

mu.1_interval = quantile(extract_pooled_alternative$mu, probs=c(0.05,0.95))
```

The posterior expectation for the the mean of the first machine is 85.73 with a 90% credible interval of 80.4557274, 90.6151465.

Hierarchical model

a)

Since we are using a hierarchical model now, we will make use of the hyperparameters μ and τ to get the paramets μ_i and σ for the machines. Each machine will have their own mean μ_i and a common variance σ.

$y\_i \sim \mathcal{N}(\mu_i, \sigma)$

$\mu_i \sim \mathcal{N}(\mu, \tau)$

$\sigma \sim \text{Inv-}\chi^2(\tau)$.

$\mu \sim \mathcal{N}(100, 10)$

$\tau \sim \text{Inv-}\chi^2(10)$

b)

```
data {
 int<lower=0> N;
```

```
 int<lower=0> J;
 vector[J] y[N];
 real mean_mu_prior;
 real<lower=0> mean_sigma_prior;
 real<lower=0> sigma_prior;
}

parameters {
 real mu;
 real <lower=0> sigma;
 real <lower=0> tau;
 vector[J] mu_i;
}

model {
  //hyperpriors
  mu ~ normal(mean_mu_prior, mean_sigma_prior);
  tau ~ inv_chi_square(sigma_prior);

 //prior
  for(j in 1:J){
    mu_i[j] ~ normal(mu, tau);
  }
  sigma ~ inv_chi_square(tau);

 //likelihoods
 for(j in 1:J){
   y[,j] ~ normal(mu_i[j], sigma);
 }

}

generated quantities {
 real ypred;
 real ypred_7;
 ypred = normal_rng(mu_i[6], sigma);
 ypred_7 = normal_rng(mu, sigma);
}
```

```
mean_mu_prior = 100
mean_sigma_prior = 10
sigma_prior = 10
hierarchical_data <- list(
  y = factory,
  N = nrow(factory),
  J = ncol(factory),
 mean_mu_prior = mean_mu_prior,
 mean_sigma_prior = mean_sigma_prior,
 sigma_prior = sigma_prior
)
```

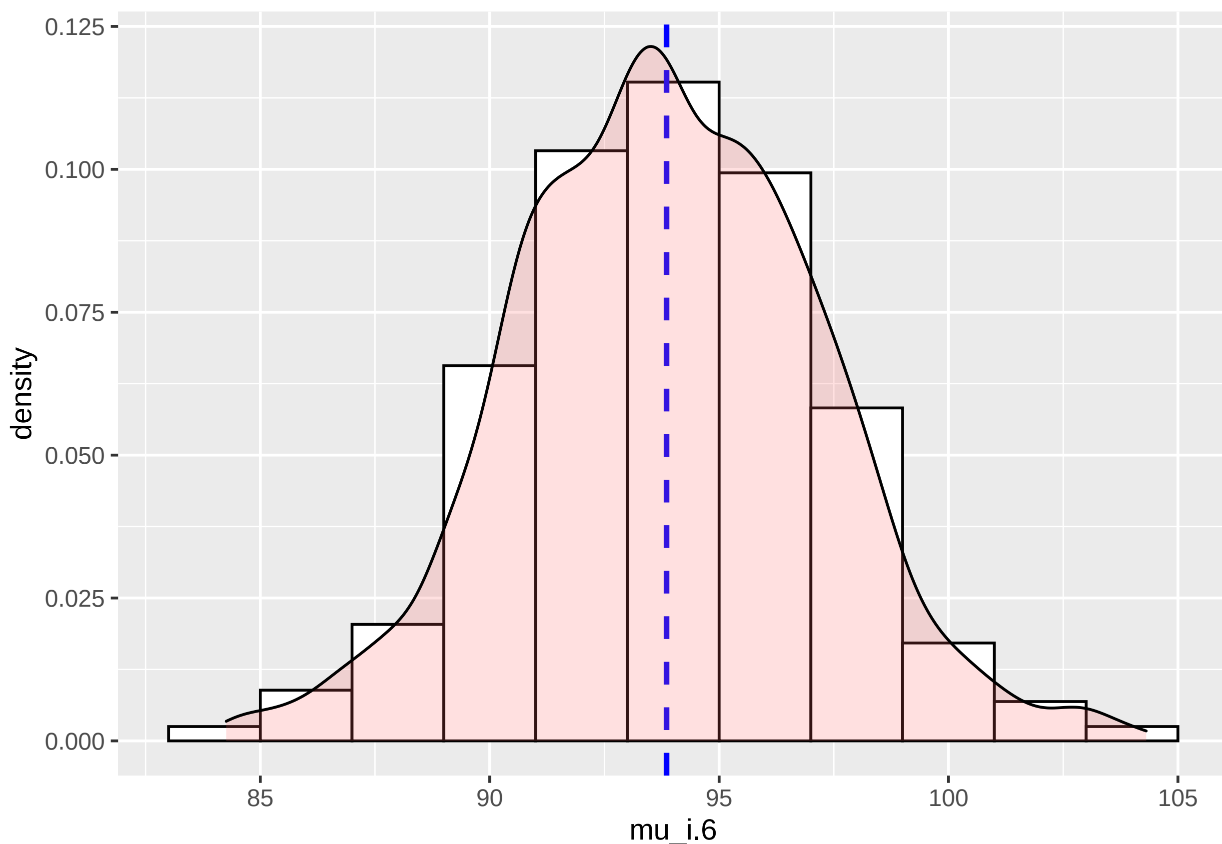
```
fit_hierarchical = sampling(hierarchicalmodel,
  data = hierarchical_data,           # named list of data
  chains = 4,             # number of Markov chains
  warmup = 1000,          # number of warmup iterations per chain
```

```
  iter = 2000,              # total number of iterations per chain
  cores = 1,                # number of cores (could use one per chain)
  refresh = 0               # no progress shown
  )
```

```
## Warning: There were 76 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
```

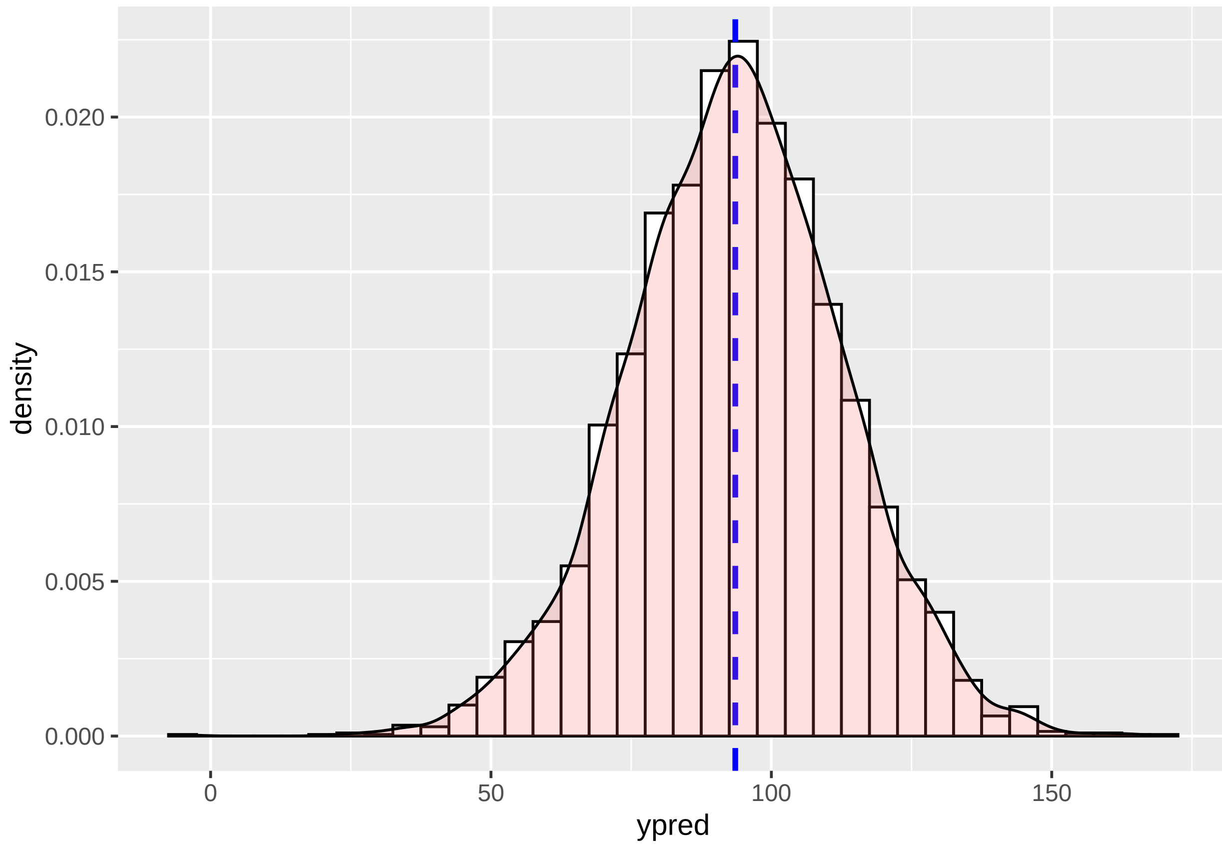
c)

```
extract_hierarchical <- data.frame(extract(fit_hierarchical))


ggplot(extract_hierarchical, aes(x=mu_i.6)) +
  geom_histogram(aes(y=..density..), binwidth = 2, colour="black", fill="white") +
  geom_vline(aes(xintercept=mean(mu_i.6)), color="blue", linetype="dashed", size=1) +
  geom_density(alpha=.2, fill="#FF6666")
```

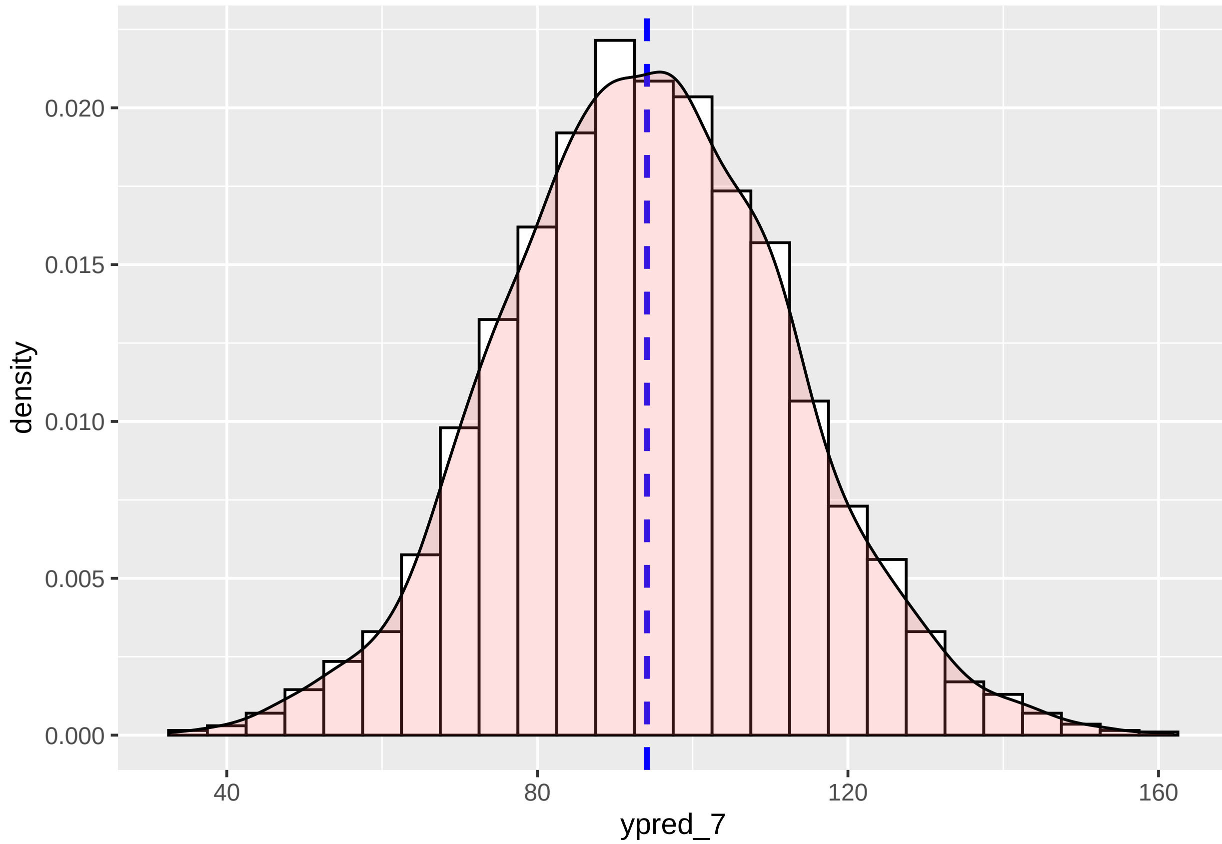



```
ggplot(extract_hierarchical, aes(x=ypred)) +
  geom_histogram(aes(y=..density..), binwidth = 5, colour="black", fill="white") +
```

```
  geom_vline(aes(xintercept=mean(ypred)), color="blue", linetype="dashed", size=1) +
  geom_density(alpha=.2, fill="#FF6666")
```




```
ggplot(extract_hierarchical, aes(x=ypred_7)) +
  geom_histogram(aes(y=..density..), binwidth = 5, colour="black", fill="white") +
  geom_vline(aes(xintercept=mean(ypred_7)), color="blue", linetype="dashed", size=1) +
  geom_density(alpha=.2, fill="#FF6666")
```



i) We can see the distribution of the mean of the 6th machine in the first histogram. The mean of the distribution lies around 94.

ii) We can see the distribution of the predictions of the 6th machine in the second histogram.

iii) The final histogram shows the posterior predicitve distribution of the 7th machine. Since we have a hierarchical distribution we have a separate model for it.

d)

```
data {
 int<lower=0> N;
 int<lower=0> J;
 vector[J] y[N];
 real mean_mu_prior;
 real<lower=0> mean_sigma_prior;
 real<lower=0> sigma_prior;
}

parameters {
 real mu;
 real <lower=0> sigma;
 real <lower=0> tau;
 vector[J] mu_i;
}
```

```
model {
  //hyperpriors
  mu ~ normal(mean_mu_prior, mean_sigma_prior);
  tau ~ gamma(sigma_prior, sigma_prior);

 //prior
  for(j in 1:J){
    mu_i[j] ~ normal(mu, tau);
  }
  sigma ~ inv_chi_square(tau);

 //likelihoods
 for(j in 1:J){
   y[,j] ~ normal(mu_i[j], sigma);
 }

}

generated quantities {
 real ypred;
 ypred = normal_rng(mu, sigma);
}
```

```r
hierarchical_data_alternative <- list(
  y = factory,
  N = nrow(factory),
  J = ncol(factory),
 mean_mu_prior = 1,
 mean_sigma_prior = 10,
 sigma_prior = 1
)

fit_hierarchical_alternative = sampling(hierarchicalmodel_alternative,
  data = hierarchical_data_alternative,            # named list of data
  chains = 4,             # number of Markov chains
  warmup = 1000,          # number of warmup iterations per chain
  iter = 2000,            # total number of iterations per chain
  cores = 1,              # number of cores (could use one per chain)
  refresh = 0             # no progress shown
  )
```

```
## Warning: There were 425 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: The largest R-hat is 1.15, 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 quant
## Running the chains for more iterations may help. See
```

```
## http://mc-stan.org/misc/warnings.html#tail-ess
```

```
extract_hierarchical_alternative <- data.frame(extract(fit_hierarchical_alternative))

mu.1_interval = quantile(extract_hierarchical_alternative$mu_i.1, probs=c(0.05,0.95))
```

The posterior expectation for the the mean of the first machine is 77.45 with a 90% credible interval of 63.6219577, 87.2625813.