BDA A7 Jupyter

2023-02-01

1. Linear model: drowning data with Stan

```
library(rstan)
```

```
## Loading required package: StanHeaders
```

```
## Loading required package: ggplot2
```

```
## rstan (Version 2.21.5, 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)
```

```
rstan_options(auto_write = TRUE)
library(aaltobda)
library('posterior')
```

```
## This is posterior version 1.3.0
```

```
##
## Attaching package: 'posterior'
```

```
## The following object is masked from 'package:aaltobda':
##
##     mcse_quantile
```

```
## The following objects are masked from 'package:rstan':
##
##     ess_bulk, ess_tail
```

```
## The following objects are masked from 'package:stats':
##
##     mad, sd, var
```

```
data("drowning")
library(ggplot2)
library(bayesplot)
```

```
## This is bayesplot version 1.9.0
```

```
## - Online documentation and vignettes at mc-stan.org/bayesplot
```

```
## - bayesplot theme set to bayesplot::theme_default()
```

```
##    * Does _not_ affect other ggplot2 plots
```

```
##    * See ?bayesplot_theme_set for details on theme setting
```

```
##
## Attaching package: 'bayesplot'
```

```
## The following object is masked from 'package:posterior':
##
##     rhat
```

```
set.seed(583)
library(cmdstanr)
```

```
## This is cmdstanr version 0.5.3
## - CmdStanR documentation and vignettes: mc-stan.org/cmdstanr
## - Use set_cmdstan_path() to set the path to CmdStan
## - Use install_cmdstan() to install CmdStan
```

Linear model with Gaussian residual model

Time as a predictor

No. drownings as target variable

i) What is the trend of the No. people drowning per year?

ii) What is the prediction for the year 2020?

a)

Original Stan code with comments on what should be fixed and how:

```
stan_broken <- "
data {
  int<lower=0> N; // number of data points
  vector[N] x; // observation year // Here we should add int and change to array
  vector[N] y; // observation number of drowned // add int and change to array
  real xpred;     // prediction year
}
parameters {
  real alpha;
  real beta;
  real<upper=0> sigma; // change upper to lower
}
transformed parameters {
  vector[N] mu = alpha + beta*x;
}
model {
  // add prior models
  y ~ normal(mu, sigma);
}
generated quantities {
  real ypred = normal_rng(mu, sigma); // change mu to predicted: alpha + beta*xpred
}
"
```

Fixed Stan code:

```
stan_program <- "
data {
  int<lower=0> N; // number of data points
  array[N] int x; // observation year
  array[N] int y; // observation number of drowned
```

```
  real xpred;     // prediction year
}
parameters {
  real alpha;
  real beta;
  real<lower=0> sigma;
}
transformed parameters {
  vector[N] mu = alpha + beta*x;
}
model {
  alpha_prior ~ normal(mu_prior_alpha, sigma);
  beta_prior ~ normal(mu_prior_beta, sigma);
  y ~ normal(mu, sigma);
}
generated quantities {
  real ypred = normal_rng(alpha + beta*xpred, sigma);
}
"

f <- write_stan_file(stan_program)
print(f)
```

```
## [1] "/tmp/RtmpBO4yOX/model_9ecc22628a96b66aea61bd8a625528cd.stan"
```

b)

99% confidence interval boundaries are 2.58 standard deviations away from the mean. Let's divide the range by 2.58*2

```
stdev <- (69*2)/(2.58*2)
stdev
```

```
## [1] 26.74419
```

Approximate numerical value for σ_β is 27

c)

I think I would normally give the values to the model in R, but as this task seems to be asking for the prior to be added in the Stan code, I guess I would change my beta prior in the model block to have the actual values instead of parameters for mu and sigma. Code would look like this:

```
stan_program <- "
data {
  int<lower=0> N; // number of data points
  array[N] int x; // observation year
  array[N] int y; // observation number of drowned
  real xpred;     // prediction year
}
parameters {
  real alpha;
  real beta;
  real<lower=0> sigma;
}
transformed parameters {
```

```
  vector[N] mu = alpha + beta*x;
}
model {
  alpha_prior ~ normal(mu_prior_alpha, sigma);
  beta_prior ~ normal(0, 27); // CHANGES MADE HERE
  y ~ normal(mu, sigma);
}
generated quantities {
  real ypred = normal_rng(alpha + beta*xpred, sigma);
}
"

f <- write_stan_file(stan_program)
print(f)
```

```
## [1] "/tmp/RtmpBO4yOX/model_7ba28f176d30fd6b4b6bd5554367c79a.stan"
```

d)

We will use a weakly informative prior for the intercept alpha. The prior should be very weakly informative to allow data to show effect and not be too bound by the expected intercept.

As we use a prior mean 0 for beta it is natural to use the provided mean of historical drownings 138 to approximate the intercept mean. The standard deviation should be very large because we want the prior to be weakly informative. The intercept value doesn't have to be a reasonable prediction for year 0, because we are not interested of year 0 but rather of recent changes. Therefore alpha can for example be negative.

Because we already have figured out a good prior standard deviation for beta, we can approximate a standard deviation for alpha by multiplying the standard deviation for beta (27) by the number of years between year 0 and our period of interest (roughly 2000)

```
27*2000
```

```
## [1] 54000
```

alpha prior: normal(138, 54000)

2. Hierarchical model: factory data with Stan

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

```
##   V1  V2  V3  V4  V5  V6
## 1 83 117 101 105  79  57
## 2 92 109  93 119  97  92
## 3 92 114  92 116 103 104
## 4 46 104  86 102  79  77
## 5 67  87  67 116  92 100
```

I'll use the following Gaussian models:

a)

Mathematical notations: Separate model

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

$\mu_j \sim N(0, 10)$

$\sigma_j \sim Inv - \chi^2(10)$

Pooled model

$y \sim N(\mu, \sigma)$

$\mu \sim N(0, 10)$

$\sigma \sim gamma(1, 1)$

Hierarchical model

$y \sim N(\mu_j, \sigma)$

$\mu_0 \sim N(0, 10)$ hyper prior

$\tau \sim gamma(1, 1)$ hyper prior

$\mu \sim N(\mu_0, \tau)$

$\sigma \sim lognormal(0, 0.5)$

Differences: In separate model, each machine has its own model, meaning that separate model has μ and σ values for each machine.

In pooled model, all measurements are combined and there is no distinction between machines meaning common μ and σ values for all machines.

The hierarchical model has separate μ values for each machine but a common σ value.

b)

```
separate_stan <- "
data {
  int<lower=0> N;
  int<lower=0> J;
  vector[J] y[N];
}

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

model {
  // 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]);
  }
}
generated quantities {
  real ypred[J];
```

```
  // Compute predictive distribution for the first machine
  for (j in 1:J){
    ypred[j] = normal_rng(mu[j], sigma[j]); // pred dist for all machines
  }
}

"

separate_file <- write_stan_file(separate_stan)
```

```
stan_data <- list(y = factory, N = nrow(factory), J = ncol(factory))
```

```
separate_model <- stan(file = separate_file, data = stan_data)
```

Separate model

```
## Trying to compile a simple C file
```

```
## Running /usr/lib/R/bin/R CMD SHLIB foo.c
## clang -flto=thin -I"/usr/share/R/include" -DNDEBUG   -I"/usr/local/lib/R/site-library/Rcpp/include/"
## In file included from <built-in>:1:
## In file included from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eigen
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1:
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:88:
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: error: unknown ty
## namespace Eigen {
## ^
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:16: error: expected
## namespace Eigen {
##                ^
##                ;
## In file included from <built-in>:1:
## In file included from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eigen
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1:
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:96:10: fatal error: 'complex' file not fou
## #include <complex>
##          ^~~~~~~~
## 3 errors generated.
## make: *** [/usr/lib/R/etc/Makeconf:168: foo.o] Error 1
##
## SAMPLING FOR MODEL 'model_da1623b4c1f03a3b6adb68c5d68c5c0a' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.2 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: 0.295552 seconds (Warm-up)
## Chain 1:                0.121887 seconds (Sampling)
## Chain 1:                0.417439 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'model_da1623b4c1f03a3b6adb68c5d68c5c0a' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 9e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.09 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: 0.307954 seconds (Warm-up)
## Chain 2:                0.140292 seconds (Sampling)
## Chain 2:                0.448246 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'model_da1623b4c1f03a3b6adb68c5d68c5c0a' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 9e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.09 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: 0.307869 seconds (Warm-up)
## Chain 3:                0.143653 seconds (Sampling)
## Chain 3:                0.451522 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'model_da1623b4c1f03a3b6adb68c5d68c5c0a' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 9e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.09 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: 0.332347 seconds (Warm-up)
## Chain 4:                0.145765 seconds (Sampling)
## Chain 4:                0.478112 seconds (Total)
## Chain 4:
```

```
separate_model
```

```
## Inference for Stan model: model_da1623b4c1f03a3b6adb68c5d68c5c0a.
## 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]      50.58    0.14  8.95   31.84   44.68   51.13   57.10   66.34  3814
## mu[2]      49.00    0.23 12.17   26.47   40.52   48.71   57.27   74.82  2924
## mu[3]      58.41    0.20 11.60   33.86   50.83   59.22   66.82   78.37  3220
## mu[4]      47.97    0.22 12.01   25.36   39.65   47.49   55.82   71.52  2891
## mu[5]      60.66    0.22 12.69   34.46   51.96   61.09   70.41   82.28  3415
## mu[6]      51.08    0.18 10.42   29.95   44.24   51.39   58.43   70.23  3176
## sigma[1]   15.92    0.06  3.40   10.27   13.43   15.60   18.06   23.38  3757
## sigma[2]   24.66    0.08  4.48   15.99   21.57   24.61   27.68   33.26  3047
## sigma[3]   16.07    0.08  4.34    8.57   12.86   15.73   18.95   25.18  3249
## sigma[4]   26.26    0.08  4.42   17.63   23.21   26.31   29.19   35.02  2742
## sigma[5]   15.76    0.08  4.75    7.62   12.31   15.59   18.94   25.52  3440
## sigma[6]   18.70    0.07  3.81   11.87   15.92   18.51   21.15   26.72  3356
## ypred[1]   50.84    0.29 18.62    9.20   39.75   52.26   63.56   83.11  4046
## ypred[2]   48.73    0.47 28.02  -10.36   30.58   50.44   68.27   99.77  3565
## ypred[3]   58.97    0.32 20.63   13.42   46.27   61.41   73.51   93.17  4049
```

```
## ypred[4]   48.09    0.51 28.27  -11.40   30.00   48.92   67.68  100.08  3118
## ypred[5]   60.56    0.36 20.88   14.02   48.06   62.93   75.62   94.80  3315
## ypred[6]   51.01    0.40 21.75    5.51   37.43   52.52   65.79   90.51  2927
## lp__     -346.87    0.06  2.34 -352.39 -348.23 -346.54 -345.15 -343.21  1587
##          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 Wed Feb  1 12:43:54 2023.
## 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_stan <- "
data {
  int<lower=0> N;
  vector[N] y;
}

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

model {
  mu ~ normal(0,10);
  sigma ~ gamma(1,1);
  y ~ normal(mu, sigma);
}

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

"
```

```
pooled_file <- write_stan_file(pooled_stan)
print(pooled_file)
```

Pooled model

```
## [1] "/tmp/RtmpBO4yOX/model_0776dec6ad76a825a7bbd9ae90536e37.stan"
```

```
stan_data <- list(N = length(c(t(factory[,1:6]))),
                  x = rep(1:length(c(t(factory[,1:6]))), 1),
                  y = c(t(factory[,1:6])))

pooled_model <- stan(file = pooled_file, data = stan_data)
```

```
## Trying to compile a simple C file
## Running /usr/lib/R/bin/R CMD SHLIB foo.c
## clang -flto=thin -I"/usr/share/R/include" -DNDEBUG   -I"/usr/local/lib/R/site-library/Rcpp/include/"
## In file included from <built-in>:1:
## In file included from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eigen
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1:
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:88:
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: error: unknown ty
## namespace Eigen {
## ^
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:16: error: expected
## namespace Eigen {
##                ^
##                ;
## In file included from <built-in>:1:
## In file included from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eigen
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1:
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:96:10: fatal error: 'complex' file not fou
## #include <complex>
##          ^~~~~~~~~
## 3 errors generated.
## make: *** [/usr/lib/R/etc/Makeconf:168: foo.o] Error 1
##
## SAMPLING FOR MODEL 'model_0776dec6ad76a825a7bbd9ae90536e37' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.7e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.17 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: 0.022424 seconds (Warm-up)
## Chain 1:                0.0174 seconds (Sampling)
## Chain 1:                0.039824 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'model_0776dec6ad76a825a7bbd9ae90536e37' 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: 0.025434 seconds (Warm-up)
## Chain 2:                0.01581 seconds (Sampling)
## Chain 2:                0.041244 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'model_0776dec6ad76a825a7bbd9ae90536e37' 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: 0.022676 seconds (Warm-up)
## Chain 3:                0.01597 seconds (Sampling)
```

```
## Chain 3:                0.038646 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'model_0776dec6ad76a825a7bbd9ae90536e37' 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: 0.023311 seconds (Warm-up)
## Chain 4:                0.015739 seconds (Sampling)
## Chain 4:                0.03905 seconds (Total)
## Chain 4:
```

```
pooled_model
```

```
## Inference for Stan model: model_0776dec6ad76a825a7bbd9ae90536e37.
## 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      85.51    0.08  3.28   78.60   83.44   85.66   87.70   91.66  1558    1
## sigma   16.07    0.05  1.87   12.92   14.73   15.89   17.17   20.22  1426    1
## ypred   85.83    0.27 16.55   54.09   74.68   85.80   97.16  118.68  3787    1
## lp__  -155.58    0.03  1.06 -158.46 -156.00 -155.25 -154.81 -154.55  1653    1
##
## Samples were drawn using NUTS(diag_e) at Wed Feb  1 12:44:39 2023.
## 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).
```

```
hierarchical_stan <- "
data {
  int<lower=0> N;
  int<lower=0> J;
  vector[J] y[N];

}

parameters {
```

```
  vector[J] mu;
  real mu_0;
  real tau;
  real sigma;
}

  model {
    // priors
    mu_0 ~ normal(0,10);
    tau ~ gamma(1,1);
    sigma ~ gamma(1,1);
    for (j in 1:J){
      mu[j] ~ normal(mu_0, tau);
}
    // likelihood
    for (j in 1:J) {
      y[,j] ~ normal(mu[j], sigma);
    }
}

generated quantities {
  real pred6;
  real pred7;

  pred6 = normal_rng(mu[6], sigma);
  pred7 = normal_rng(mu_0, tau);
}

"

hierarchical_file <- write_stan_file(hierarchical_stan)
```

```
stan_data <- list(N=dim(factory)[1],
                  J=dim(factory)[2],
                  y=factory)
hierarchical_model <- stan(file = hierarchical_file, data = stan_data)
```

Hierarchical model

```
## Trying to compile a simple C file
```

```
## Running /usr/lib/R/bin/R CMD SHLIB foo.c
## clang -flto=thin -I"/usr/share/R/include" -DNDEBUG   -I"/usr/local/lib/R/site-library/Rcpp/include/"
## In file included from <built-in>:1:
## In file included from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eigen
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1:
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:88:
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: error: unknown ty
## namespace Eigen {
## ^
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:16: error: expected
## namespace Eigen {
##                ^
##                ;
```

```
## In file included from <built-in>:1:
## In file included from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eigen
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1:
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:96:10: fatal error: 'complex' file not fou
## #include <complex>
##          ^~~~~~~~
## 3 errors generated.
## make: *** [/usr/lib/R/etc/Makeconf:168: foo.o] Error 1
##
## SAMPLING FOR MODEL 'model_5dd7a3d085fd9624362ccdcd93627b49' NOW (CHAIN 1).
## Chain 1: Rejecting initial value:
## Chain 1:   Error evaluating the log probability at the initial value.
## Chain 1: Exception: normal_lpdf: Scale parameter is -0.857866, but must be > 0!  (in 'model231128598
##
## Chain 1: Rejecting initial value:
## Chain 1:   Error evaluating the log probability at the initial value.
## Chain 1: Exception: normal_lpdf: Scale parameter is -0.358429, but must be > 0!  (in 'model231128598
##
## Chain 1: Rejecting initial value:
## Chain 1:   Error evaluating the log probability at the initial value.
## Chain 1: Exception: normal_lpdf: Scale parameter is -0.228198, but must be > 0!  (in 'model231128598
##
## Chain 1:
## Chain 1: Gradient evaluation took 1.6e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.16 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: 0.101848 seconds (Warm-up)
## Chain 1:                0.081969 seconds (Sampling)
## Chain 1:                0.183817 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'model_5dd7a3d085fd9624362ccdcd93627b49' NOW (CHAIN 2).
## Chain 2: Rejecting initial value:
## Chain 2:   Error evaluating the log probability at the initial value.
## Chain 2: Exception: normal_lpdf: Scale parameter is -0.236746, but must be > 0!  (in 'model231128598
##
## Chain 2:
## Chain 2: Gradient evaluation took 9e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.09 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: 0.115977 seconds (Warm-up)
## Chain 2:                0.079209 seconds (Sampling)
## Chain 2:                0.195186 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'model_5dd7a3d085fd9624362ccdcd93627b49' NOW (CHAIN 3).
## Chain 3: Rejecting initial value:
## Chain 3:   Error evaluating the log probability at the initial value.
## Chain 3: Exception: normal_lpdf: Scale parameter is -0.331034, but must be > 0!  (in 'model2311285983
##
## Chain 3: Rejecting initial value:
## Chain 3:   Error evaluating the log probability at the initial value.
## Chain 3: Exception: normal_lpdf: Scale parameter is -0.663584, but must be > 0!  (in 'model2311285983
##
## Chain 3: Rejecting initial value:
## Chain 3:   Error evaluating the log probability at the initial value.
## Chain 3: Exception: normal_lpdf: Scale parameter is -1.41451, but must be > 0!  (in 'model231128598
##
## Chain 3: Rejecting initial value:
## Chain 3:   Error evaluating the log probability at the initial value.
## Chain 3: Exception: normal_lpdf: Scale parameter is -0.51649, but must be > 0!  (in 'model231128598
##
## Chain 3: Rejecting initial value:
## Chain 3:   Error evaluating the log probability at the initial value.
## Chain 3: Exception: normal_lpdf: Scale parameter is -1.61712, but must be > 0!  (in 'model231128598
##
## Chain 3:
## Chain 3: Gradient evaluation took 1e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.1 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: 0.201412 seconds (Warm-up)
## Chain 3:                0.092842 seconds (Sampling)
## Chain 3:                0.294254 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'model_5dd7a3d085fd9624362ccdcd93627b49' NOW (CHAIN 4).
## Chain 4: Rejecting initial value:
## Chain 4:   Error evaluating the log probability at the initial value.
## Chain 4: Exception: normal_lpdf: Scale parameter is -1.92633, but must be > 0!  (in 'model2311285982
##
## Chain 4:
## Chain 4: Gradient evaluation took 9e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.09 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: 0.140064 seconds (Warm-up)
## Chain 4:                0.086272 seconds (Sampling)
## Chain 4:                0.226336 seconds (Total)
## Chain 4:

## Warning: There were 66 divergent transitions after warmup. See
## https://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
## https://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
## https://mc-stan.org/misc/warnings.html#tail-ess
```

```
hierarchical_model
```

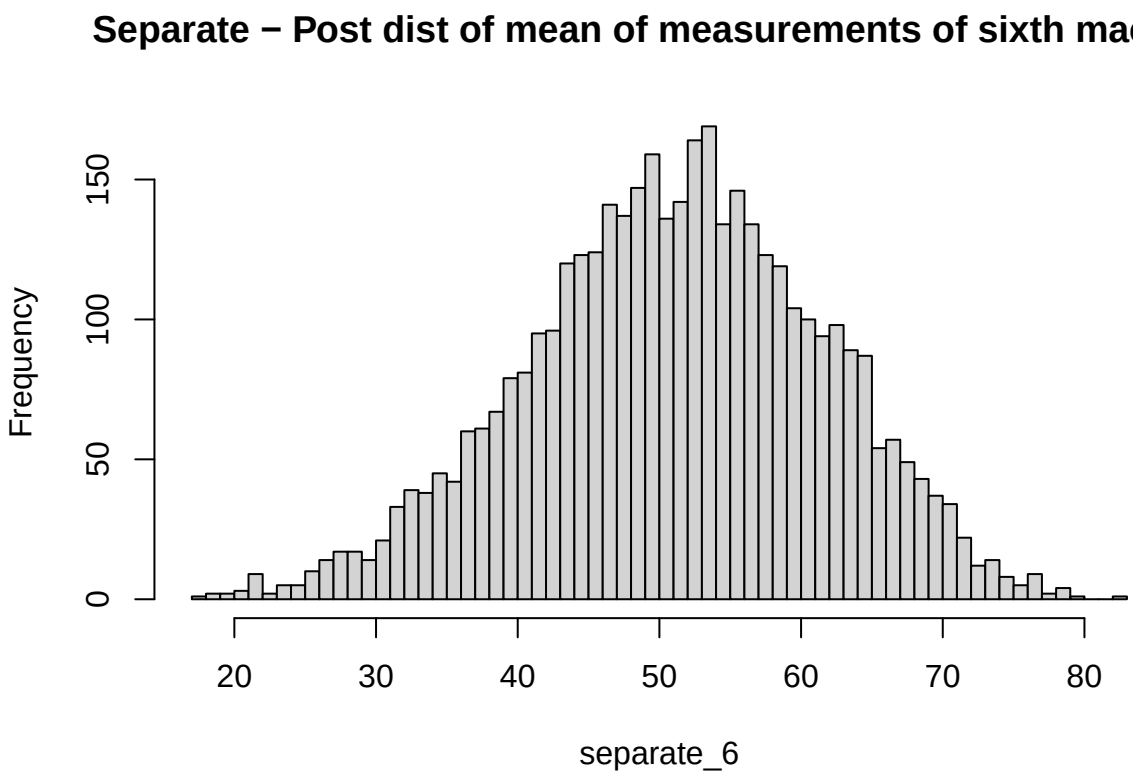
```
## Inference for Stan model: model_5dd7a3d085fd9624362ccdcd93627b49.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean    sd    2.5%     25%     50%     75%   97.5% n_eff Rhat
## mu[1]   75.71    0.12  5.24   65.47   72.13   75.73   79.24   85.51  1850 1.00
## mu[2]  100.18    0.26  5.86   88.05   96.31  100.43  104.29  110.76   494 1.00
## mu[3]   85.11    0.09  5.10   75.19   81.64   85.17   88.58   95.08  3594 1.00
## mu[4]  104.49    0.37  6.49   89.76  100.52  104.89  108.89  116.25   310 1.01
## mu[5]   86.96    0.09  5.23   76.19   83.70   86.88   90.40   97.25  3239 1.00
## mu[6]   83.66    0.08  5.15   73.12   80.33   83.83   87.06   93.57  4061 1.00
## mu_0    66.79    0.41 12.42   40.93   58.12   67.83   76.78   85.84   928 1.00
## tau     14.09    0.25  5.47    4.22   10.05   13.85   17.85   24.87   472 1.00
## sigma   12.52    0.06  1.56    9.92   11.39   12.38   13.49   15.84   678 1.00
## pred6   83.76    0.22 13.82   56.87   74.44   83.86   92.56  111.66  3784 1.00
## pred7   66.95    0.42 19.33   23.11   55.87   70.31   81.20   95.85  2101 1.00
## lp__  -171.15    0.08  2.31 -176.46 -172.42 -170.81 -169.45 -167.69   937 1.00
##
## Samples were drawn using NUTS(diag_e) at Wed Feb  1 12:45:27 2023.
## 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)

Separate model

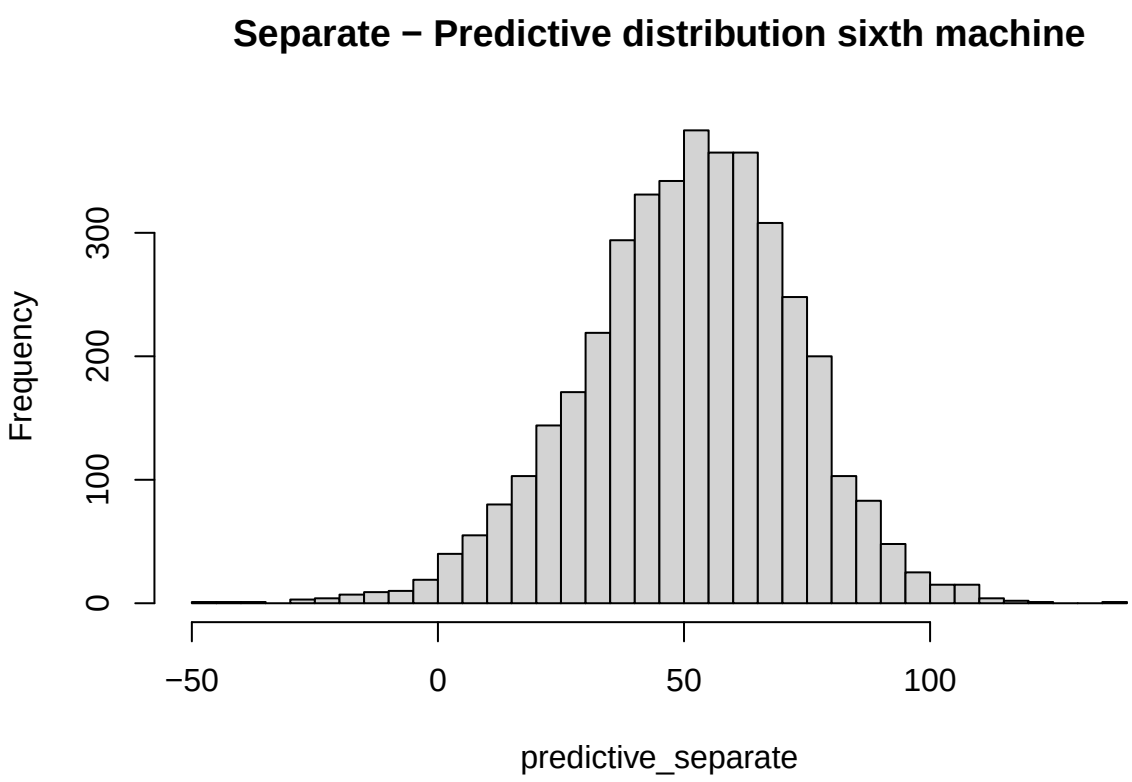
i) Results for the separate seem to be a bit lower than they might have been with some different prior specifications, I have specified my priors to be consistent with d), for example using inv chi square specifications for sigma could have increased the measurements.

```
separate_6 <- as.data.frame(separate_model)[,6]

hist(separate_6, breaks = 50,
     main="Separate - Post dist of mean of measurements of sixth machine")
```



ii) .

```
predictive_separate <- as.data.frame(separate_model)[,18]

hist(predictive_separate, breaks=50, main="Separate - Predictive distribution sixth machine")
```

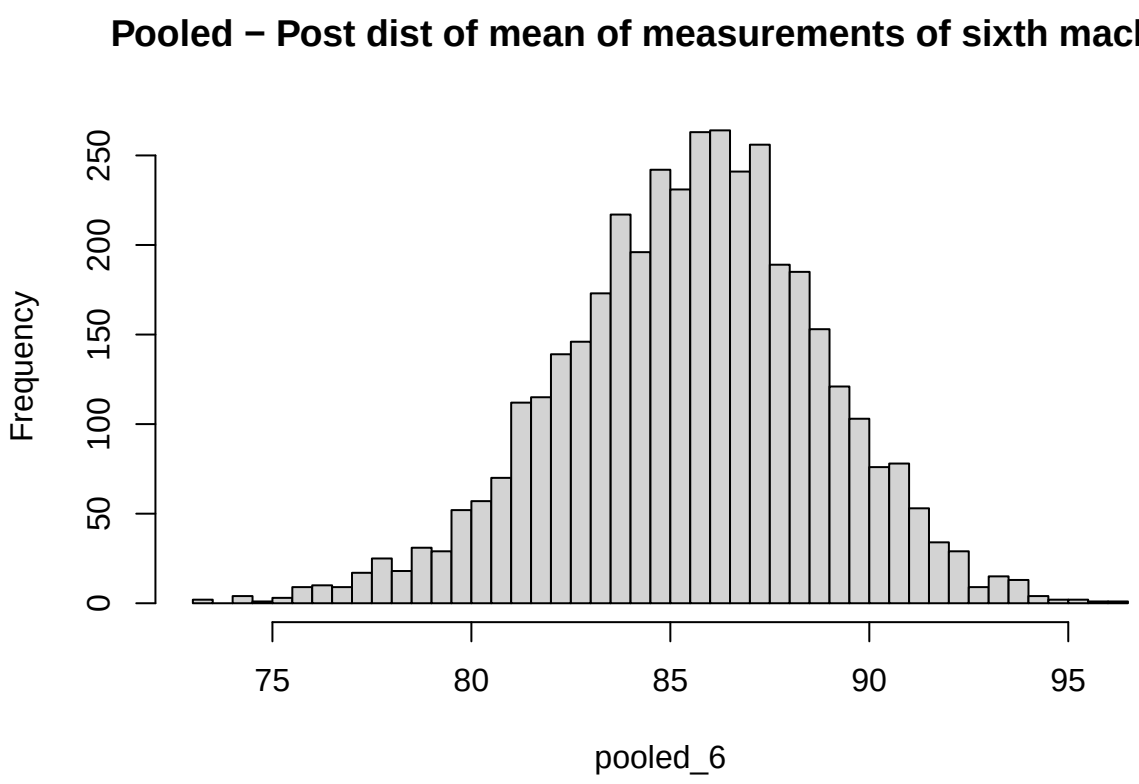


iii) We don't have enough information to answer this question for the separate model. The only related information we have is the prior. For pooled and hierarchical models we will be able to get more informed predictions.

Pooled model

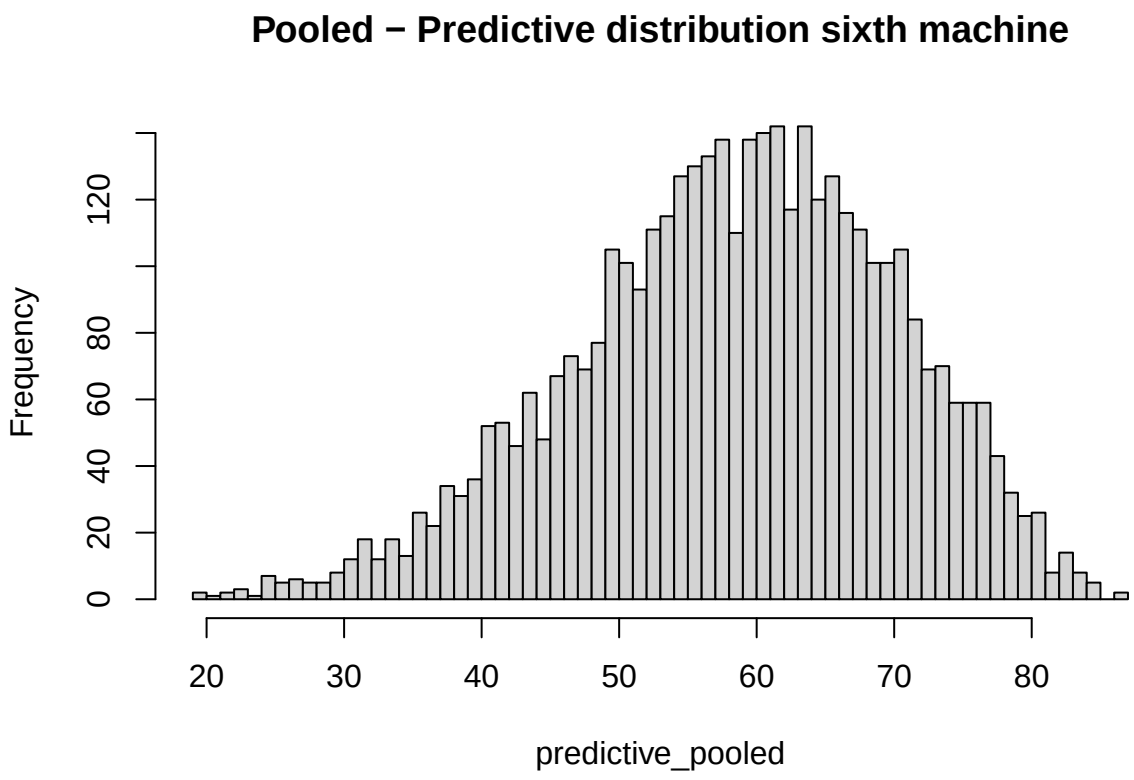
i) .

```
pooled_6 <- as.data.frame(pooled_model)$mu

hist(pooled_6, breaks = 50,
     main="Pooled - Post dist of mean of measurements of sixth machine")
```



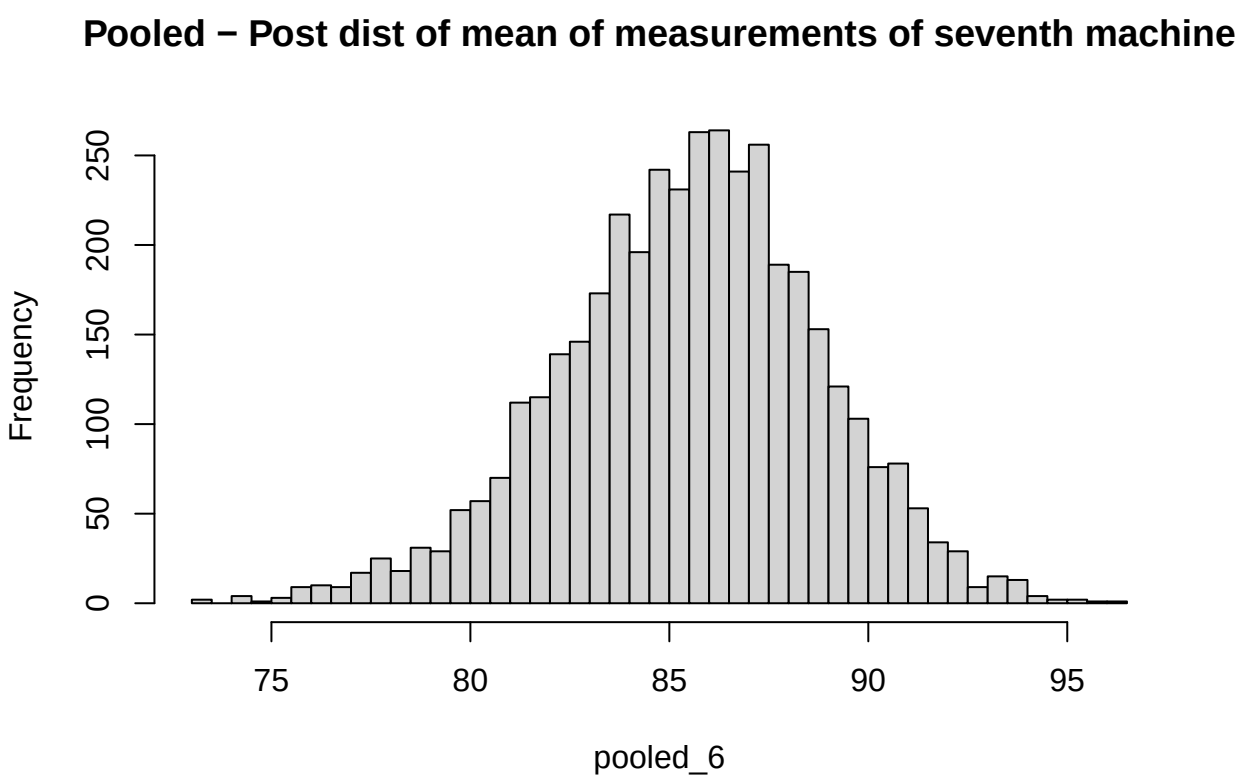
ii) .

```
predictive_pooled <- as.data.frame(separate_model)[,3]


hist(predictive_pooled, breaks=50, main="Pooled - Predictive distribution sixth machine")
```



iii) .

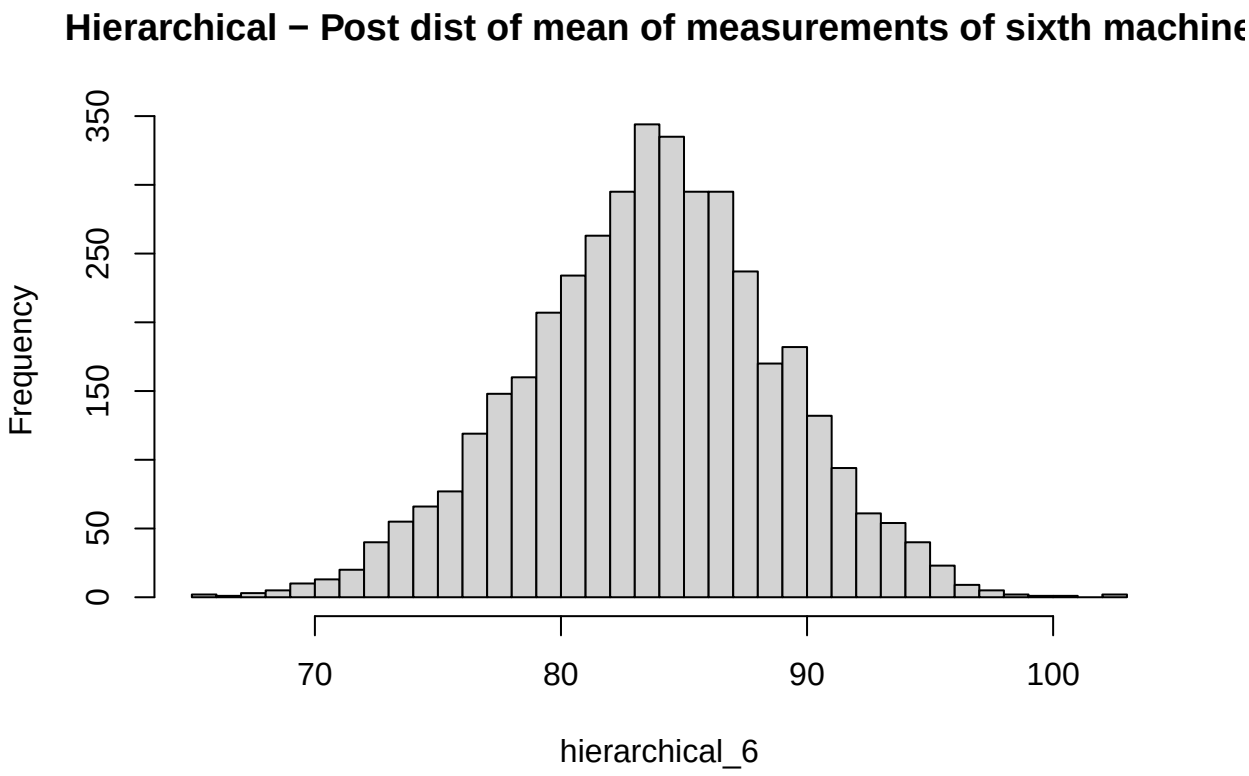
```
hist(pooled_6, breaks = 50,
     main="Pooled - Post dist of mean of measurements of seventh machine")
```



Hierarchical model

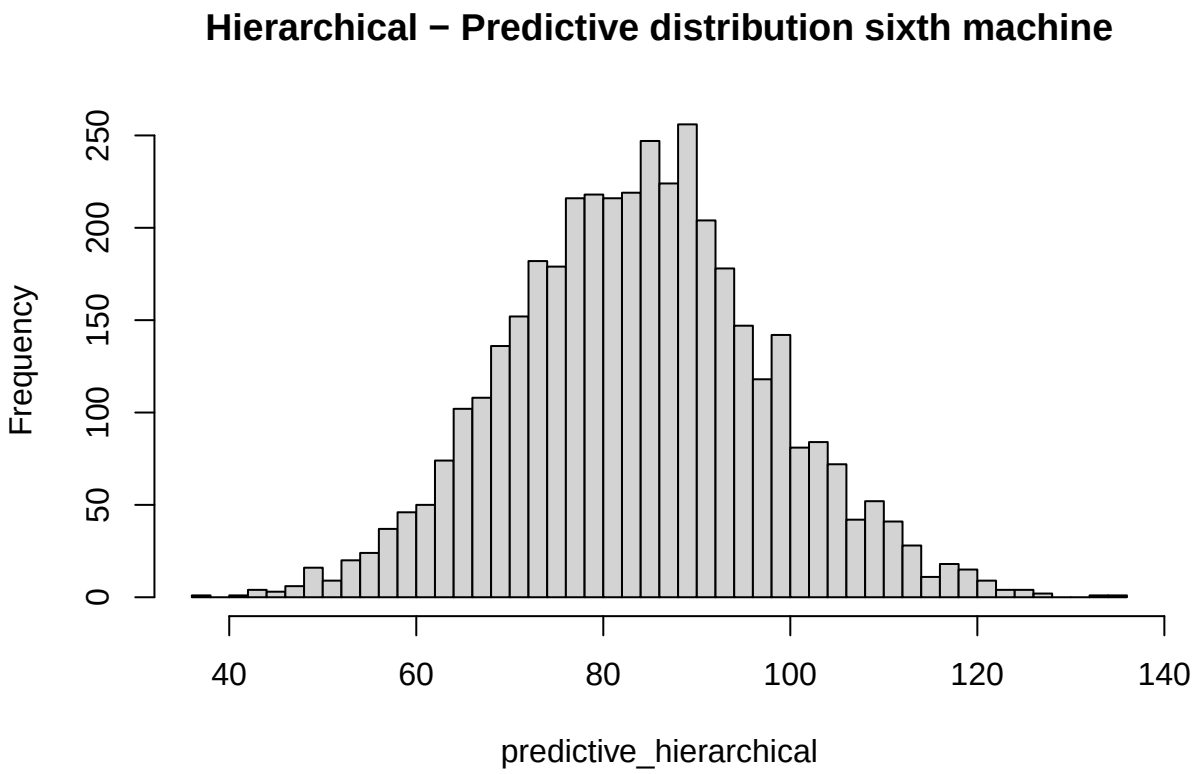
i) .

```
hierarchical_6 <- as.data.frame(hierarchical_model)[,6]

hist(hierarchical_6, breaks = 50,
     main="Hierarchical - Post dist of mean of measurements of sixth machine")
```



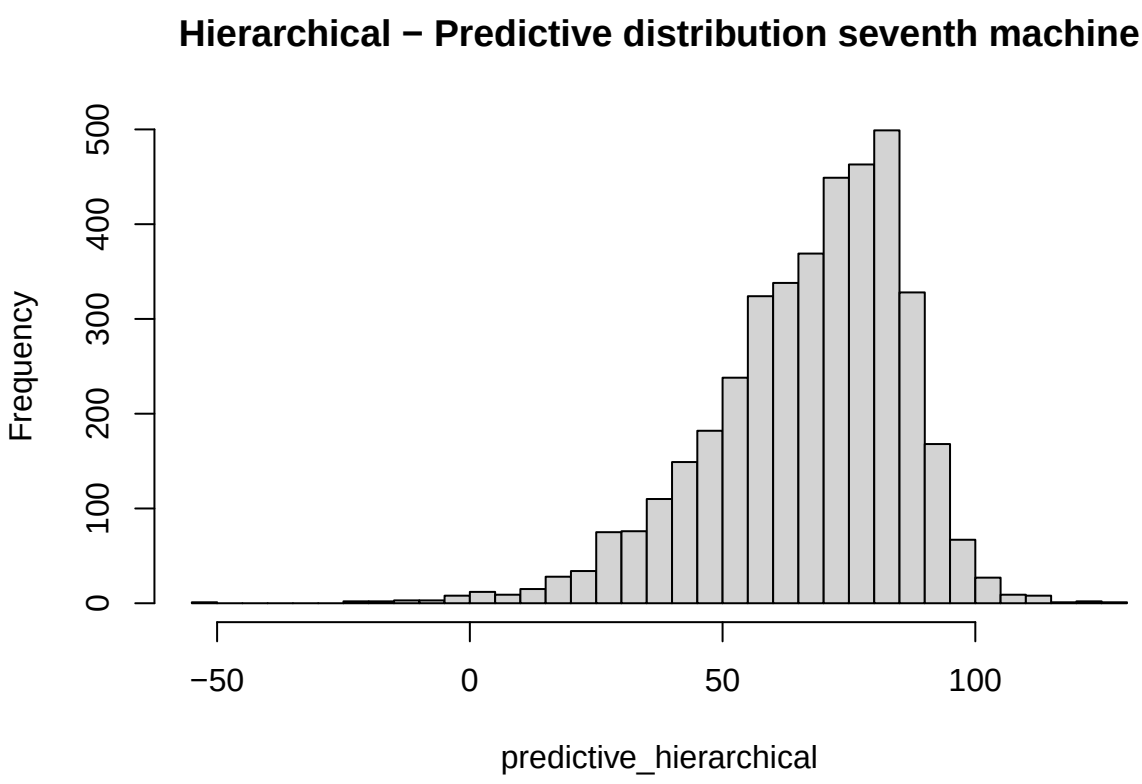
ii) .

```
predictive_hierarchical <- as.data.frame(hierarchical_model)[,10]

hist(predictive_hierarchical, breaks=50, main="Hierarchical - Predictive distribution sixth machine")
```



iii) .

```
predictive_hierarchical <- as.data.frame(hierarchical_model)[,11]

hist(predictive_hierarchical, breaks=50, main="Hierarchical - Predictive distribution seventh machine")
```



Here especially my prior specifications seem to spread the distribution so that it is strongly skewed to the left.

d)

The specifications have already been in adapted in step b).

```
separate_quantiles <- as.data.frame(separate_model)[,1]
quantile(separate_quantiles, c(0.05,0.95))
```

Separate model 90% credible interval

```
##       5%      95%
## 34.90664 64.20180
```

```
pooled_quantiles <- as.data.frame(pooled_model)[,1]
quantile(pooled_quantiles, c(0.05,0.95))
```

Pooled model 90% credible interval

```
##       5%      95%
## 79.94242 90.72520
```

```
hierarchical_quantiles <- as.data.frame(hierarchical_model)[,1]
quantile(hierarchical_quantiles , c(0.05,0.95))
```

Hierarchical model 90% credible interval

```
##       5%      95%
## 67.09732 84.08715
```