BSDA: Assignment 7

October 15, 2023

```
knitr::opts_chunk$set(echo = TRUE)
setwd("~/Desktop/stan-demo-files")
library(rstan)
library(bsda)
```

Times used for reading and self-study exercise: 10
Time used for the assignment: 12
Good with the assignment:
Good that we got to work with the Stan code especially for the Hierarchical model.
Things to improve in the assignment:
I would have preferred for us to have one extra lecture on the syntax behind Stan. I believe it is easier to interpret the output if one understand the syntax and its process.

Question 1: Linear model, drowning data with Stan

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

Q1a, Finding mistakes and correcting the Stan code

For subtask Q1a we will need to find three different mistakes and correct them. First, we will locate and report the mistakes and then we will include the full corrected Stan in the report.

For this fitted linear model we use year as the predictor, say x, and number of drownings as the target variable denoted as y. I found three different mistakes in the Stan code. One mistake was located in the parameter block, another in the model block and the last in the generated quantities block of the Stan code.

The first mistake, that could be located in the parameters block, was that the parameter sigma was defined with the upper bound set to 0, ie that sigma only takes on negative real numbers. However, sigma should instead be defined as a real positive number. Hence, we correcting this mistake by doing the following:

```
real <upper=0> sigma; -->  real <lower=0> sigma;
```

The second located mistake was when computing the linear model without semicolon at the end of the line which can be corrected by adding the semicolon at the end.

```
y ~ normal(mu, sigma) -->  y ~ normal(mu, sigma);
```

The third and last mistake was found in the generated quantities block. We want to predict the number of drownings, ypred, using the xpred as the predictor thus we need to redefine the ypred from

```
real ypred = normal_rng(mu, sigma);
```

to having

```
ypred = normal_rng(alpha + beta*xpred, sigma);
```

Now when the three mistakes has been located and corrected below we report the full corrected Stan code and checking if the corrected code is syntactically correct or not by using rstudio stanc-function from the rstan package.

```
data {
  int<lower=0>N;  // number of data points
  vector[N]x;     // observation year
  vector[N]y;     // observation number of drowned
  real xpred;     // prediction year, 2020
}

parameters {
  real alpha;
  real beta;
  real <lower=0> sigma;
}

transformed parameters {
  vector[N] mu = alpha + beta*x;
}

model {
  y ~ normal(mu, sigma);
}

generated quantities {
  real ypred = normal_rng(alpha + beta*xpred, sigma);
}
```

```
rstan:::rstudio_stanc("Assign7.stan")
```

and when checking the full corrected Stan code I get that the code is syntactically correct.

Q1b, Determine a weakly-informative prior for β

For Q1b we will determine a suitable weakly-informative prior distribution $N(0, \sigma_\beta)$ for the slope parameter β and we will determine a value of σ_β so it satisfies $P(-69 < \beta < 69) = 0.99$.

Hence, we have defined that β comes from the Gaussian distribution with the population mean corresponding to 0 and an undefined σ_β. We have calculated that $\sigma_\beta \approx 26.796$ satisfies the condition above and has been calculated according to below:

$\beta \sim N(0, \sigma_\beta)$

$P(-69 < \beta < 69) = 0.99$

$P(\beta < 69) - P(\beta < -69) = 0.99$

$P(\frac{\beta-0}{\sigma_\beta} < \frac{69-0}{\sigma_\beta}) - P(\frac{\beta-0}{\sigma_\beta} < \frac{-69-0}{\sigma_\beta}) = 0.99$

$P(Z < \frac{69}{\sigma_\beta}) - P(Z < \frac{-69}{\sigma_\beta}) = 0.99$

$\Phi(\frac{69}{\sigma_\beta}) - \left[1 - \Phi(\frac{69}{\sigma_\beta})\right] = 0.99$

$\Phi(\frac{69}{\sigma_\beta}) = \frac{1.99}{2} = 0.995$

$\Phi(\frac{69}{\sigma_\beta}) \approx \Phi(2.575) \rightarrow \frac{69}{\sigma_\beta} \approx 2.575 \rightarrow \sigma_\beta \approx \frac{69}{2.575} \approx 26.796$

Q1c, Adding the prior into the Stan code

Now when the σ_β has been obtained to be $\sigma_\beta = 26.796$, we can define the prior distribution for β as

$\beta \sim N(0, \sigma_\beta)$,

where $E[\beta] = 0$ and $V[\beta] = \sigma_\beta^2 = 26.796^2$

As has been stated in the Assignment when no prior has been defined in for the parameter then on default in Stan an uniform prior is being used. To obtained the desire prior $\beta \sim N(0, \sigma_\beta = 26.796)$ in Stan code I added line

```
beta ~ normal(mu_0, sigma_0)
```

in the model block of the Stan code. Thus, the model block looks the following after adding the desired prior:

```
model {
  beta ~ normal(mu_0, sigma_0)
  y ~ normal(mu, sigma);
}
```

where μ_0 and σ_0 has been defined in the data block of the Stan code. The Stan code is still syntactically correct.

Q1d, Adding a weakly informative prior for the intercept α

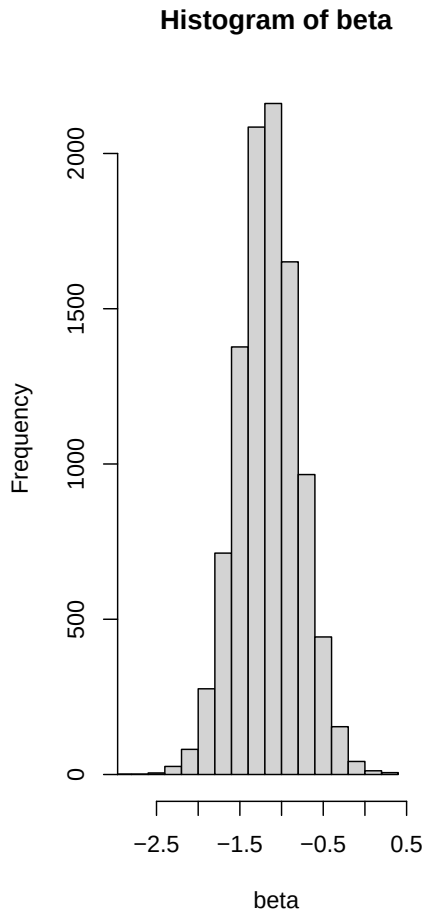
Now when the weakly informative prior distribution for β has been included in the Stan code we want to add another weakly informative prior for the intercept α in the

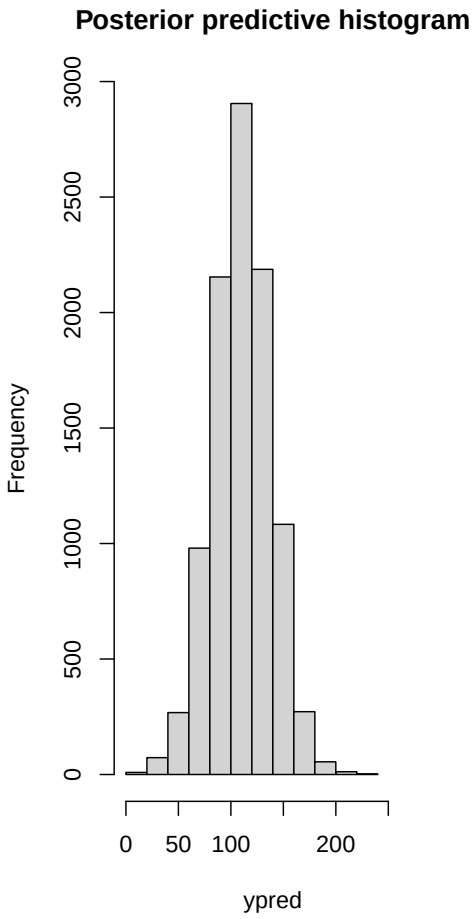
code. We added another line in the model block to include the prior for intercept α and the line looks the following:

```
alpha ~  normal(mu_intercept, sigma_intercept)
```

The mean and standard deviation of the prior distribution is being defined in the data block of the Stan code. We decide to let the mean correspond to the mean value of the drownings i.e $E[\alpha] = \bar{y}$. However, we want to use a fixed value of for standard deviation σ_α. We set σ_α to a large and fixed value for weakly informative reasons hence we set $\sigma_\alpha = 60$.

```
attach(drowning)
data_list <- list(N = nrow(drowning),
                  x = year,
                  y = drownings,
                  xpred = 2020,
                  mu_0 = 0,
                  sigma_0 = 26.796,
                  mu_intercept = mean(drowning$drownings),
                  sigma_intercept = 60)

fit_model <- stan(file = "Assign7.stan",
                  data = data_list,
                  iter = 5000,
                  chains = 4)

samples <- rstan::extract(fit_model, permute = FALSE)

 par(mfrow = c(1, 2))
hist(samples[,,2],
     xlim = c(-2.825289, 0.7),
     xlab = "beta",
     main = "Histogram of beta")

hist(samples[,,44],
     xlim = c(0, 280),
     main = "Posterior predictive histogram",
     xlab = "ypred") # good
```





Question 2: Models using factory data with Stan

For Q2 we will do three different multilevel models - separate, pooled and hierarchical model - based on the factory data in the bsda package which we are getting access to by running this code

```
set.seed(123)
data("factory")
attach(factory)
```

Separated Model

For the Separated model we separate the data into group, in this case we separate into $J = 6$ groups. In this model the groups are being sampled independently from J separated models, i.e $(y_{i,1}, ..., y_{i,J})$ have separated parameter such that $(\theta_1, ..., \theta_J) \forall i$. When assuming that $y_{i,j} \sim N(\cdot) \forall (i,j)$, $\mu_j \sim N(\cdot)$ and $\sigma_j \sim \text{Inv}-\chi^2(\cdot)$ then the separated model can be summarized mathematically as:

$$
\begin{aligned}
y_{ij} &\sim N\left(\mu_j, \sigma_j\right) \\
\mu_j &\sim N(0, \sigma_j) \\
\sigma_j &\sim \text{Inv}-\chi^2(\cdot)
\end{aligned}
$$

For the prior distribution of σ_j we have that the degrees of freedom corresponds to $v_j = N + J - 1 = 10 \forall j = 1, ..., J$. We want to use a weakly informative prior for μ_j thus we will use a normal distribution around mean equal to 0 and a large standard deviation which gives us $\mu_j \sim N(0, 100)$. The separated model will looks like the following:

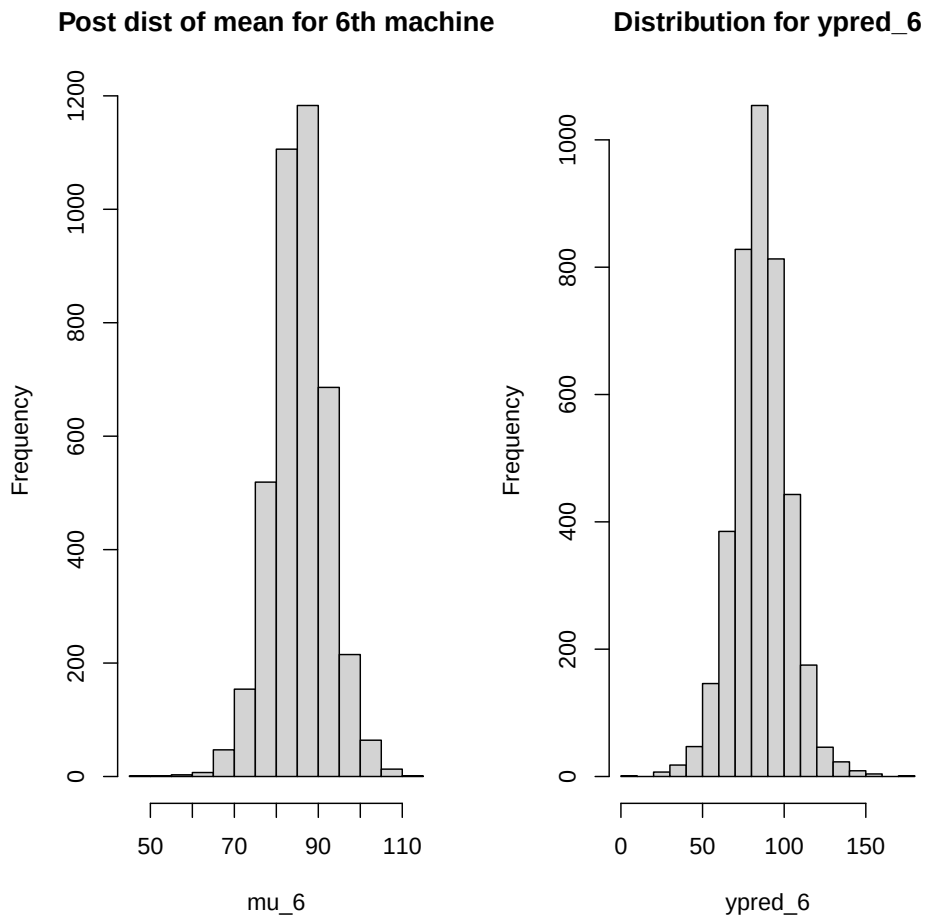
$$
\begin{aligned}
y_{ij} &\sim N\left(\mu_j, \sigma_j\right) \\
\mu_j &\sim N(0, 100) \\
\sigma_j &\sim \text{Inv}-\chi^2(10)
\end{aligned}
$$

Now when the separated model has been stated and mathematically denoted with weakly informative priors we need to create a Stan code which computes the model. Further, based on the Stan code we will plot the histogram for the μ_6 and the predictive value $\tilde{y}_6$. The stan code has the following structure

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

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

model {
  // prior distribution
  for (j in 1:J){
    mu[j] ~ normal(0,1);
    sigma[j] ~ inv_chi_square(10);
  }
  // likelihood function
  for (j in 1:J){
    y[,j] ~ normal(mu[j], sigma[j]);
  }
}

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

and the computed values from Stan code result in the following two histograms. First histogram for posterior distribution of μ_6 and the second histogram for $\tilde{y}_6$.

```
rstan:::rstudio_stanc("SeperatedModel.stan")

sm <- rstan::stan_model(file = "SeperatedModel.stan")

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

model <- rstan::sampling(sm, data = stan_data)

sampling <- rstan::extract(model, permuted = F)

par(mfrow = c(1, 2))
# Posterior distribution of the mean for the sixth machine
hist(sampling[,,6],
     main = "Post dist of mean for 6th machine",
     xlab = "mu_6")

# Predictive distribution for quality control measurements
# for sixth machine
hist(sampling[,,18],
     main = "Distribution for ypred_6",
     xlab = "ypred_6")
```



Pooled Model / Joint model

For the pooled model we have that the observations shares a common parameter element $\theta : 1x1$ or parameter vector $\boldsymbol{\theta} : Jx1$, i.e that $(y_{i,1}, ..., y_{i,J})\, \forall i$ all shares one or several parameters $\boldsymbol{\theta}$.

In our case we have the the parameter vector consist of two parameter element set to $\boldsymbol{\theta} = (\mu, \sigma) : 2x1$ and from a Bayes standpoint, in our case, the μ and σ will be attached to a weakly informative prior distribution, separately.

We assume for the observations $(y_{i,1}, ..., y_{i,J})$ to be normally distributed and for the prior distributions to be weakly informative. Hence, the pooled model can be summarized mathematically as

$$\begin{aligned} y_i &\sim N\left(\mu, \sigma\right), \forall i \\ \mu &\sim N(0, \sigma) \\ \sigma &\sim \text{Inv} - \chi^2(10) \end{aligned}$$

For the prior distribution regarding σ we have that the degrees of freedom also in this

case corresponds to $v = N + J - 1 = 10$. Further, for deciding the prior distribution for μ we want for the prior to be normally distributed around mean 0 and with a large uncertainty / deviation thus we set $\sigma_\mu = 100$ also for this model. We get the following structure for the pooled model:

$$\begin{aligned} y_i &\sim N(\mu, \sigma), \forall i \\ \mu &\sim N(0, 100) \\ \sigma &\sim \text{Inv} - \chi^2(10) \end{aligned}$$

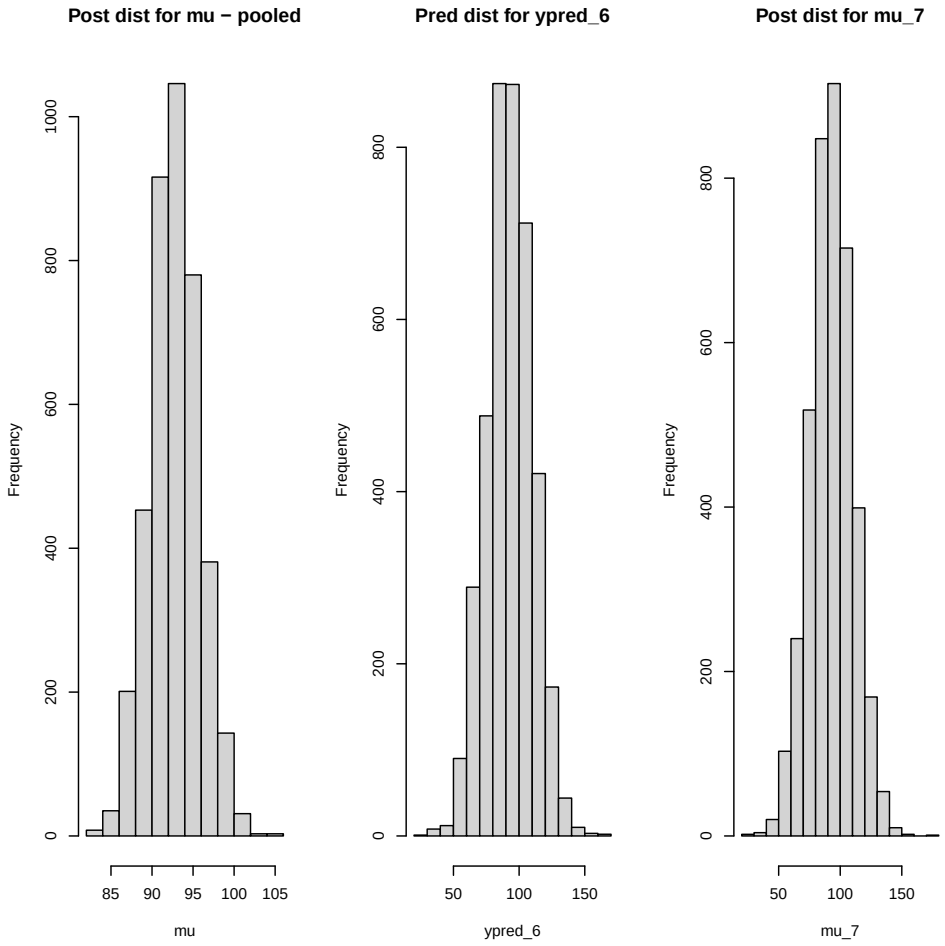
As stated above, the difference between $y_{i,k}$ and $y_{i,j}$ when $j \neq k$ is due to the sampling variation. This due to the $y_{i,k}$ and $y_{i,j}$ is being generated from the same parameters μ and σ and therefore also being generated from the same prior distributions.

To be able to compute this pooled model given the stated distributions above we create a Stan code that computes this. Further, we will visualize the distribution of μ_6, $\tilde{y}_6$ and the μ_7 using histograms where $\tilde{y}$ denotes the predictive distribution for another quality measurement of the sixth machine. The Stan code looks like the following:

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

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

model {

  mu ~ normal(0,100);
  sigma  ~ inv_chi_square(10);

  for (j in 1:J) {

    y[,j] ~ normal(mu, sigma);
  }
}

generated quantities {
  vector[J] ypred;
  real mu_new;
  for (j in 1:J) {
  ypred[j] = normal_rng(mu, sigma);

  }
  mu_new = normal_rng(mu, sigma);
}
```

Now for the histogram and computations of the model. As stated above we will create three different histograms based on the output of the stan code.

```
rstan:::rstudio_stanc("PooledModel.stan")

sm2 <- rstan::stan_model(file = "PooledModel.stan")

model2 <- rstan::sampling(sm2, data = stan_data)

sampling2 <- rstan::extract(model2, permuted = F)

par(mfrow = c(1,3))
# Posterior distribution for mean of quality measurements of sixth machine;
hist(sampling2[,,1],
     main = "Post dist for mu - pooled",
     xlab = "mu")

# Posterior distribution for y_pred_6;
hist(sampling2[,,8],
     main = "Pred dist for ypred_6",
     xlab = "ypred_6")

# Posterior distribution for mu_new where mu_new
# is the mean parameter for the seventh machine
hist(sampling2[,,9],
     main = "Post dist for mu_7",
     xlab = "mu_7")
```



When looking at the three histograms above we can see that all histograms are centered around the same value which is somewhere around 90 however their mean standard error and standard deviation tend to differ between μ_6, $\tilde{y}_6$ and μ_7. The reason why for small deviation in mean is due to the only variation included in the pooled model is the sample variation and this is being generated by a large σ in the prior distribution.

Hierarchical model

Hierarchical model can be seen a model that combines the separate and the pooled model, also called partial pooling model. In a hierarchical model the parameters, $\boldsymbol{\theta} = (\mu, \sigma) : 2x1$ in our case, is viewed a sample from one or several hyperparameters, $\boldsymbol{\phi} = (\mu_p, \sigma_p) : 2x1$. Further the hyperparameters are being assigned hyperpiors.

Let $y_{i,j} \forall (i,j)$ be an observation for which μ_j and σ_j decides the data generating process of $y_{i,j}$. Further, we let $\boldsymbol{\theta}_j = (\mu_j, \sigma_j)$ be generated from hyperpriors μ_p and

σ_p. Hence, Hierarchical model's general framework for a given group j is

$$
\begin{aligned}
&y_j \mid \theta_j, \phi \sim P\left(y_j \mid \theta_j, \phi\right) \\
&\theta_j \mid \phi \sim P\left(\theta_j \mid \phi\right) \\
&\phi \sim P(\phi)
\end{aligned}
$$

In our model we want to use weakly informative priors and hyperpriors. Because we let the $\boldsymbol{\theta}$ be a sample from $\boldsymbol{\phi}$ we need to state the hyperpriors. We assume that μ_p and σ_p comes from the normal and inverse-χ^2 distribution respectively. Since we want to use weakly informative hyperpriors we let set a large value of σ_{μ_p} and letting μ_p be centered around zero. Since, $\boldsymbol{\theta}$ is being generated by $\mu_p \sim N(\cdot)$ and $\sigma_p \sim Inv{-}\chi^2(10)$ we assume that following distributions for elements in $\boldsymbol{\theta}$ and for $y_{i,j}$:

$$
\begin{aligned}
&\mu_p \sim \mathcal{N}(0, 100) \\
&\sigma_p \sim Inv - \chi^2(10) \\
&\theta_j \mid \mu_P, \sigma_P^2 \sim \mathcal{N}\left(\mu_P, \sigma_P^2\right) \\
&y_{ij} \mid \theta_j \sim \mathcal{N}\left(\theta_j, \sigma_j^2\right)
\end{aligned}
$$

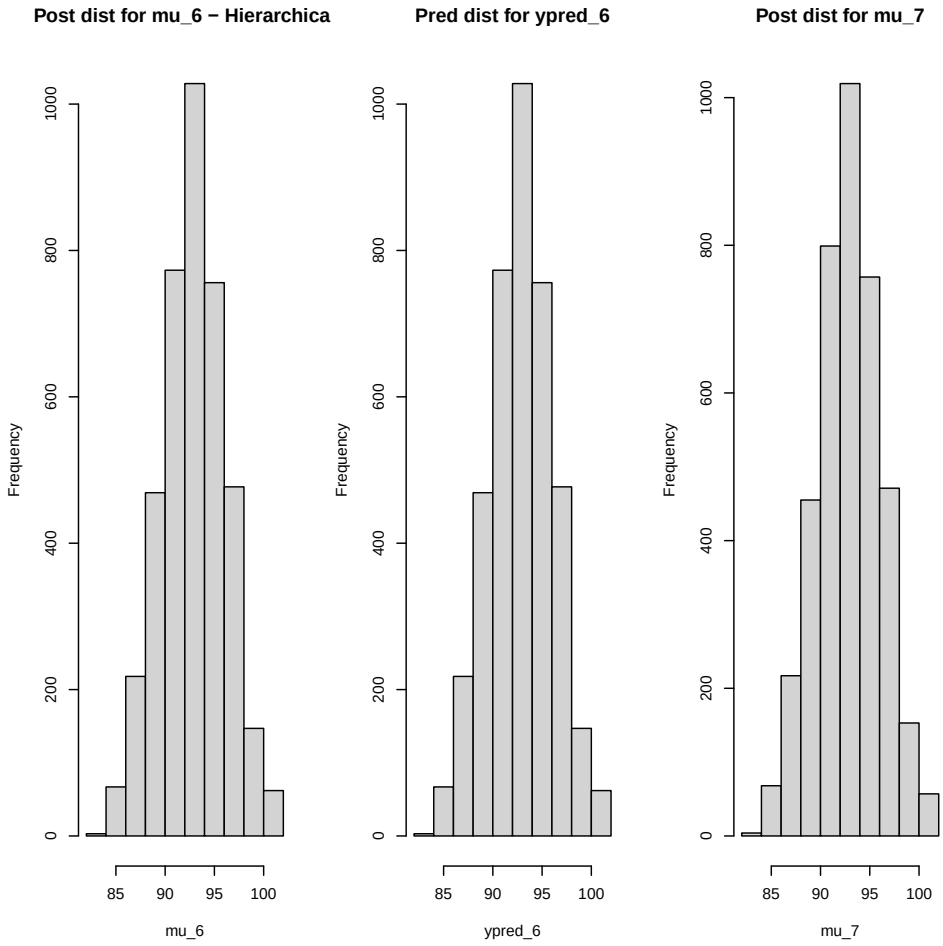
Now when the different stages for the model's framework as well as all priors has been stated we need to create a Stan code that computes the model and the code looks like

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

parameters {
  // parameters
  vector[J] mu;
  real<lower = 0> sigma;
  // hyperparameters
  real mu_p;
  real<lower=0>sigma_p;
}

model {
  //  hyperprior distributions for mu_p and sigma_p
  mu_p ~ normal(0,100);
  sigma_p  ~ inv_chi_square(10);

  // prior distribution for mu and sigma
  sigma ~ inv_chi_square(10);
  for (j in 1:J) {
   mu[j] ~ normal(mu_p, sigma_p);

  // likehood function
  y[,j]  ~ normal(mu[j], sigma);
  }
}
```

```
generated quantities {
  vector[J] ypred;
  real mu_7;
  for (j in 1:J) {
  ypred[j] = normal_rng(mu[j], sigma);
  }
  mu_7 = normal_rng(mu_p, sigma_p);
}
```

Also for this model, like for the pooled model we will visualize the output with three different histograms. Histogram for μ_6, $\tilde{y}_6$ and μ_7 separately where $\tilde{y}_j$ denotes the predictive value of the j:th machine.

```
sm3 <- rstan::stan_model(file = "HierchModel.stan")

model3 <- rstan::sampling(sm3, data = stan_data)

sampling3 <- rstan::extract(model3, permuted = F)

par(mfrow = c(1,3))
# Posterior distribution for mu_6
hist(sampling3[,,6],
     main = "Post dist for mu_6 - Hierarchical",
     xlab = "mu_6")

# Predictive distribution for ypred_6
hist(sampling3[,,6],
     main = "Pred dist for ypred_6",
     xlab = "ypred_6")

# Posterior distribution for mu_7
hist(sampling3[,,16],
     main = "Post dist for mu_7",
     xlab = "mu_7")
```



d, Reporting posterior expectation for μ_1 with 90% credible interval

For this subtask Q2d we will compute the posterior expectation of μ_1 with 90% credible interval for the three different models, i.e for separate, pooled and hierarchical model separately. The Stan codes for respective model will be used in this part. However, we will need to change the prior distribution for the pooled and separate model and the hyperprior distribution for the hierarchical model.

For separate model and pooled model we will change $\mu \sim N(0, 100)$ and $\sigma \sim inv - \chi^2(10)$ to $\mu \sim N(0, 10)$ and $\sigma \sim \Gamma(1, 1)$ respectively. For hierarchical model, the same change will be done but for the hyperpriors μ_p and σ_p.

The number of digits reported for the 90% credible interval for the posterior expectation for μ_1 is based on its Monte Carlo standard error, MCSE. A more in depth description can be found in the instructions for Assignment 7.

```
set.seed(123)

# Creating stan objects for the three different models
```

```r
stan_sep <- rstan::stan_model(file = "SeparatedPrior.stan") # Separated model
stan_pool <- rstan::stan_model(file = "PooledPrior.stan") # Pooled model
stan_hierch <- rstan::stan_model(file = "HierchPrior.stan") # Hierarchical model

# Stan sampling for the three different models
model_sep <- rstan::sampling(stan_sep, data = stan_data)
model_pool <- rstan::sampling(stan_pool, data = stan_data)
model_hierch <- rstan::sampling(stan_hierch, data = stan_data)
```

```r
# Extract value from the stan sampling, output is an array
sampling_sep <- extract(model_sep, permute = FALSE)
sampling_pool <- extract(model_pool, permute = FALSE)
sampling_hierch <- extract(model_hierch, permute = FALSE)

# Creating an object for probs argument in
# quantile() and mcse_quantile()
x <- c( (1-0.9)/2, 1-(1-0.9)/2)

# Monte carlo standard error for the three different models
# Separated model
mcse_quantile(sampling_sep[,,1], prob = 0.05)
```

```
##        mcse
## 1 0.3539865
```

```r
mcse_quantile(sampling_sep[,,1], prob = 0.95)
```

```
##        mcse
## 1 0.2076823
```

```r
# Pooled model
mcse_quantile(sampling_pool[,,1], prob = 0.05)
```

```
##        mcse
## 1 0.1583636
```

```r
mcse_quantile(sampling_pool[,,1], prob = 0.95)
```

```
##         mcse
## 1 0.09928173
```

```r
# Hierarchical model
mcse_quantile(sampling_hierch[,,1], prob = 0.05)
```

```
##        mcse
## 1 0.1905627
```

```r
mcse_quantile(sampling_hierch[,,1], prob = 0.95)
```

```
##        mcse
## 1 0.1416752
```

```r
# 90% Credible interval for the Separated model
round(quantile(sampling_sep[,,1], probs = x), digits = 0)
```

```
##  5% 95%
##  35  64
```

```r
# 90% Credible interval for the Pooled model
c( round(quantile(sampling_pool[,,1], probs = x[1]), digits = 0),
   round(quantile(sampling_pool[,,1], probs = x[2]), digits = 1))
```

```
##   5%  95%
## 80.0 90.6
```

```r
# 90% Credible interval for the Hierarchical models
round(quantile(sampling_hierch[,,1], probs = x), digits = 0)
```

```
##  5% 95%
##  67  88
```

The posterior expectation of for $\mu_1 = \mu$ with 90% credible interval when using separated, pooled and Hierarchical model corresponds to $[35, 64]$, $[80.0, 90.5]$ and $[67, 87]$ respectively. For the Pooled model we have that the μ, and σ, determines the data generating process for $y_{i,j} \forall (i, j)$ thus for Pooled model we have that $\mu_1 = \mu$.

Differences and similarities between the models

There is some differences and similarities with the three different models. The first common component for all models is that one or several parameters determines the generating data process for observation $y_{i,j}$ however the data generating process differs between the models. For separate model we have that all the observations in all groups is being generated by independent parameters while for pooled the there is a common parameter determining observations for all groups. Further, hierarchical model can be seen as a combination of the pooled and separate.