BSDA: Assignment 7

Anonymous student

Contents

General Information to include

- **Time used for reading and self-study exercises**: ~ 8 hours.
- **Time used for the assignment**: ~12 hours
- **Good with assignment**: It was good to see how hierarchical models are implemented in Stan
- **Things to improve in the assignment**: It was a really difficult and demanding assignment, I felt like only with what we saw in the lecture wasn't enough to solve the whole assignment.

1. Linear model: drowning data with Stan

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

i) What is the trend of the number of people drowning per year? (We would plot the histogram of the slope of the linear model.)

ii) What is the prediction for the year 2020? (We would plot the histogram of the posterior predictive distribution for the number of people drowning at $\tilde{x} = 2020$.)

To access the data, use:

```
library(bsda)
library(tigerstats)
library(rjson)
library(rstan)
library(lubridate)
library(ggplot2)
library(tidyverse)
library(knitr)
library(kableExtra)
library(posterior)
library(bayesplot)
```

```r
library(bayestestR)
data("drowning")
drowning<- drowning
```

Corresponding Stan code is provided in Listing 1. However, it is not entirely correct for the problem. First, there are *three mistakes*. Second, there are no priors defined for the parameters. In Stan, this corresponds to using uniform priors.

Your tasks are the following:

a) Find the three mistakes in the code and fix them. Report the original mistakes and your fixes clearly in your report. Include the *full* corrected Stan code in your report.

I am reporting all the modifications to the Stan Code using code chunks in this report. I also created a Stan file code `assignment7_1.stan` with all these modifications which I use in the `stan()` function.

This is the corrected Stan code:

```
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
}
parameters {
    real alpha;
    real beta;
    real<lower=0> sigma; // fixed upper to lower, sigma can only take values>0
}
transformed parameters {
    vector[N] mu = alpha + beta*x;
}
model {
    y ~ normal(mu, sigma); // fixed by adding ;
}
generated quantities {
    real ypred = normal_rng(alpha + beta*xpred, sigma); //fixed using xpred instead of x
}
```

b) Determine a suitable weakly-informative prior $\text{normal}(0, \sigma_\beta)$ for the slope `beta`. It is very unlikely that the mean number of drownings changes more than 50 % in one year. The approximate historical mean yearly number of drownings is 138. Hence, set σ_β so that the following holds for the prior probability for `beta`: $\Pr(-69 < \texttt{beta} < 69) = 0.99$. Determine suitable value for σ_β and report the approximate numerical value for it.

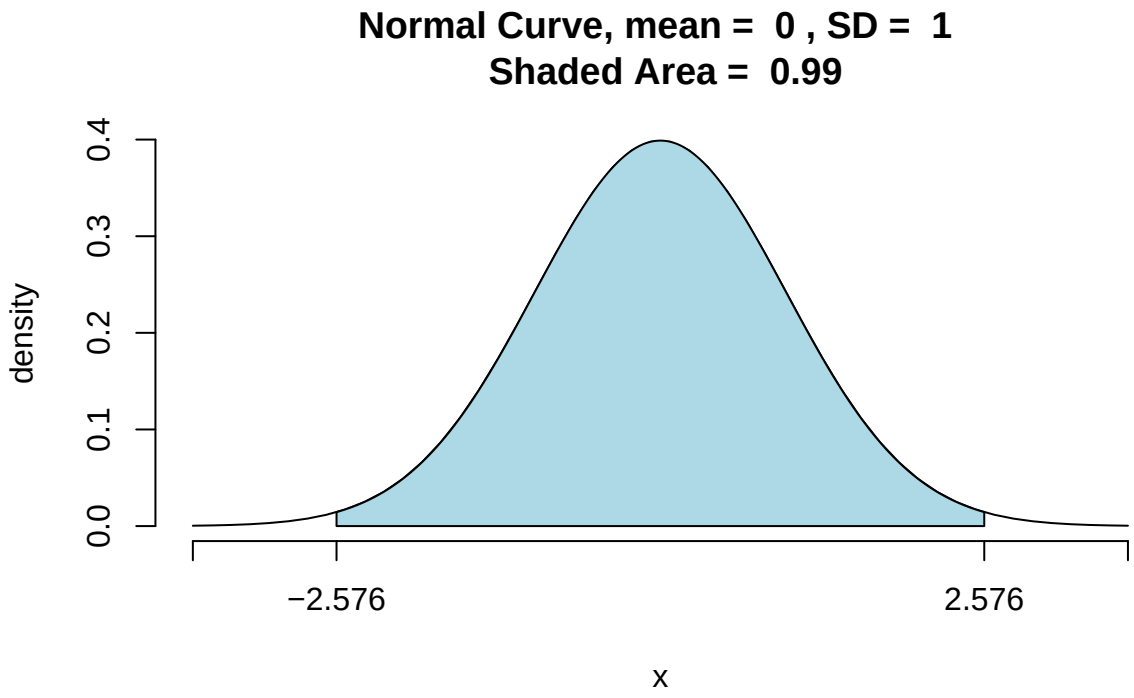
Given that we are using a weakly-informative prior for the slope $\beta \sim N(0, \sigma_\beta)$ and that we know that the historical mean yearly drownings is 138 and it doesn't varies more than 50% in a year. We can set the following condition for sigma $\Pr(-69 < \texttt{beta} < 69) = 0.99$. Then we can normalize

$$\left(\frac{-69-\mu_\beta}{\sigma_\beta} < \frac{\beta-\mu_\beta}{\sigma_\beta} < \frac{69-\mu_\beta}{\sigma_\beta}\right) = 0.99$$
$$\left(\frac{-69-0}{\sigma_\beta} < Z < \frac{69-0}{\sigma_\beta}\right) = 0.99$$

Because of symmetry in the normal distribution we have:

```
z = round(qnorm(0.005,mean=0,sd=1), 3)

pnormGC(c(z,-z),region="between",mean=0,
        sd=1,graph=TRUE)
```




```
## [1] 0.9900049
```

Then solving for σ_β we get:

$$\frac{-69}{\sigma_\beta} = -2.576 \Rightarrow \sigma_\beta = 26.787$$

Therefore, a suitable value for σ_β could be $\sigma_\beta = 27$.

Using the obtained σ_β, add the desired prior in the Stan code. In the report, in a separate section, indicate clearly how you carried out your prior implementation, e.g. "Added line ... in block ...''

I added the following line in the model section and marked it with ***:

```
beta ~ normal(0, 27);
```

```
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
}
parameters {
    real alpha;
    real beta;
    real<lower=0> sigma; // fixed upper to lower, sigma can only take values>0
}
transformed parameters {
    vector[N] mu = alpha + beta*x;
}
model {
    y ~ normal(mu, sigma); // fixed by adding ;
    beta ~ normal(0, 27); // *** added the beta prior here
}
generated quantities {
    real ypred = normal_rng(alpha + beta*xpred, sigma); //fixed using xpred instead of x
}
```

d) In a similar way, add a weakly informative prior for the intercept `alpha` and explain how you chose the prior.

I used the data mean and sd to choose a weakly informative prior for the intercept.

```
summary(drowning$drownings)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    83.0   119.5   134.0   134.3   144.8   200.0
```

```
sd(drowning$drownings)
```

```
## [1] 28.48441
```

By looking at these statistics we can choose a prior for the intercept centered about the data mean number of drownings and have most of its mass between 83 and 200. Therefore we can choose a weakly informative prior like $\alpha \sim N(134, 55)$

I added the following line in the model section and marked it with ***:

```
alpha ~ normal(134, 35);
```

```
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
}
parameters {
    real alpha;
    real beta;
    real<lower=0> sigma; // fixed upper to lower, sigma can only take values>0
}
transformed parameters {
```

```
    vector[N] mu = alpha + beta*x;
}
model {
    y ~ normal(mu, sigma); // fixed by adding ;
    beta ~ normal(0, 27); // added the beta prior here
    alpha ~ normal(134, 35); // *** added the alpha prior here
}
generated quantities {
    real ypred = normal_rng(alpha + beta*xpred, sigma); //fixed using xpred instead of x
}
```

Now we can run our model

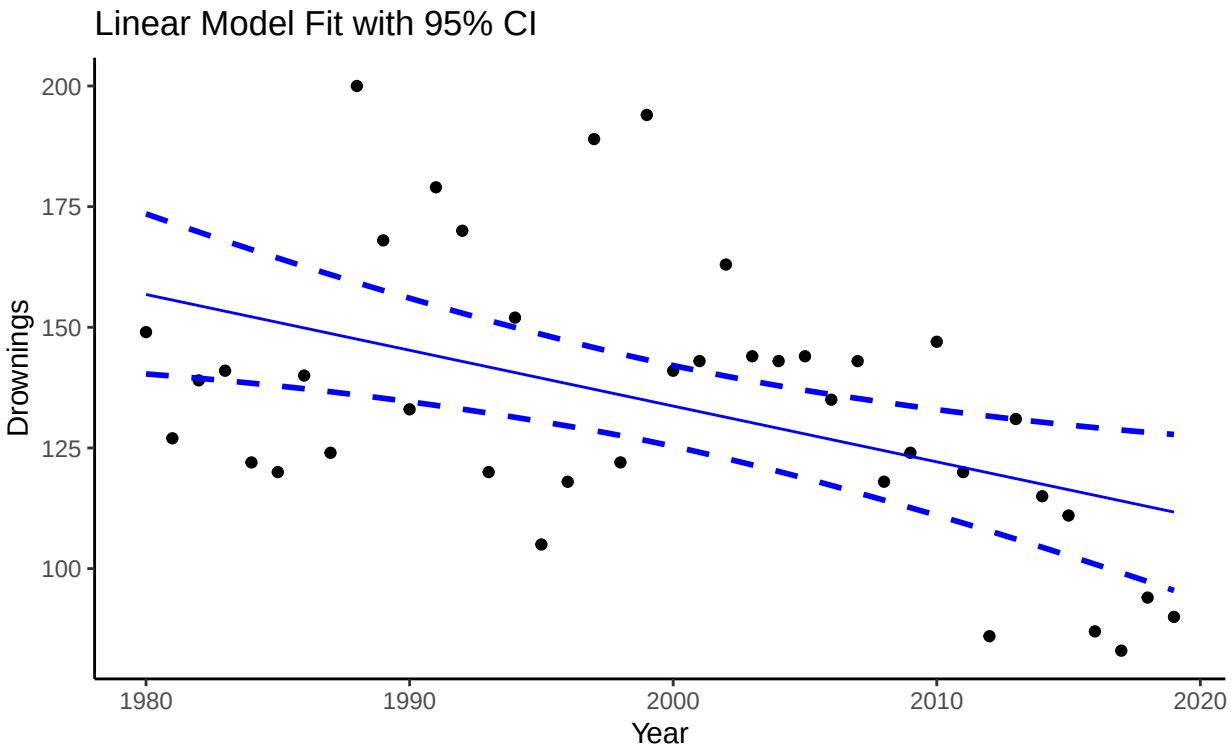
```
options(mc.cores = parallel::detectCores())

fit1 <- stan(file = 'assignment7_1.stan',
             data = list(N = nrow(drowning),
                         x = drowning$year - mean(drowning$year),
                         y = drowning$drownings,
                         xpred = 2020 - mean(drowning$year)
                         ),
             seed = 123
             )
```

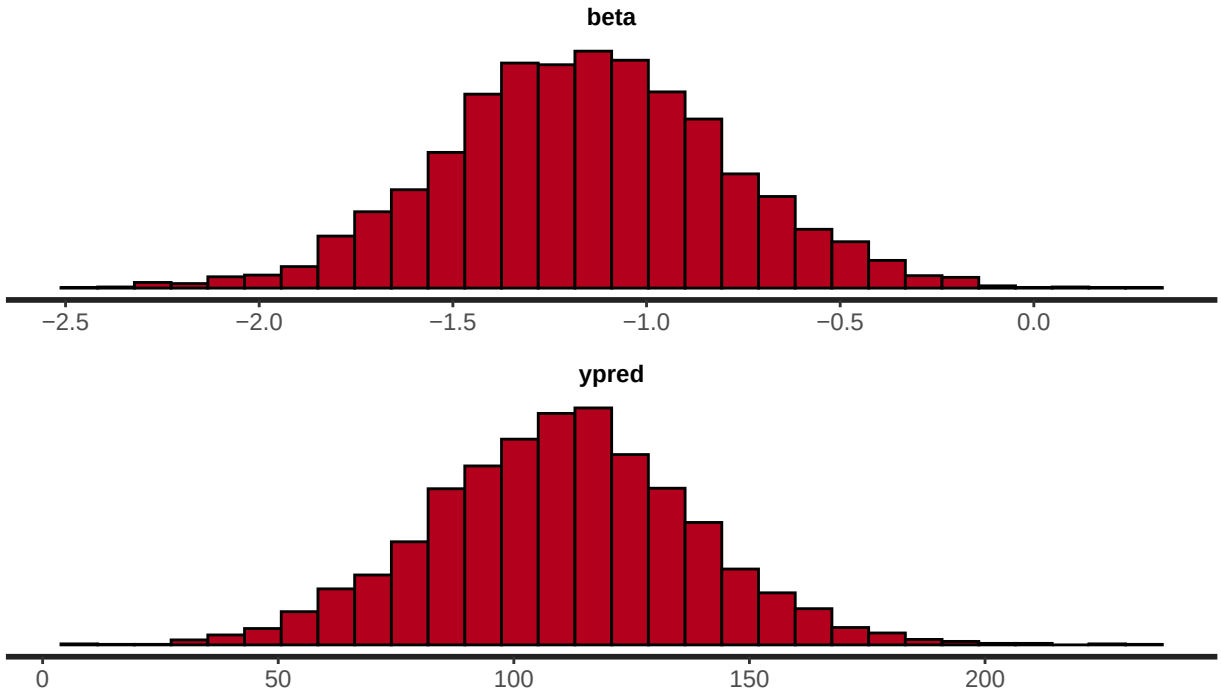
```
df_posterior <- bayestestR::describe_posterior(fit1, ci = 0.95, test = c()) %>%
  as_tibble() %>%
  filter(str_detect(Parameter, "mu")) %>%
  select(Parameter, Median, CI_low, CI_high)
df_posterior <- cbind(df_posterior, drowning)

summary_fit1 <- summary(fit1)
alpha_est <- summary_fit1$summary["alpha", "mean"]
beta_est <- summary_fit1$summary["beta", "mean"]

ggplot(df_posterior, aes(x = year, y = drownings)) +
  geom_point() + geom_line(aes(x = year,
                               y = alpha_est + beta_est * (year-mean(year))),
                           color = "blue") +
  geom_smooth(aes(y = CI_low),
              method = "loess",formula = "y ~ x", se = FALSE,
              color = 'blue', linetype = 2) +
  geom_smooth(aes(y = CI_high),
              method = "loess",formula = "y ~ x", se = FALSE,
              color = 'blue', linetype = 2) +
  labs(x = "Year", y = "Drownings", title = "Linear Model Fit with 95% CI") +
  theme_classic()
```




```
stan_hist(fit1, pars = c('beta', 'ypred'), nrow = 2)
```



2. Hierarchical model: factory data with Stan

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

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

For this problem, you'll use the following Gaussian models:

- a separate model, in which each machine has its own model/parameters
- a pooled model, in which all measurements are combined and there is no distinction between machines
- a hierarchical model, which has a hierarchical structure as described in BDA3 Section 11.6.

As in the model described in the book, use the same measurement standard deviation σ for all the groups in the hierarchical model. In the separate model, however, use separate measurement standard deviation σ_j for each group j. You should use weakly informative priors for all your models.

a) Describe the model with mathematical notation (as is done for the separate model above). Also describe in words the difference between the three models.

Separate model: this model considers each of the machines separately, building a separate model for each of them. In this case, each machine has its own μ_j and σ_j parameters.

$$
\begin{aligned}
y_{ij} &\sim N(\mu_j, \sigma_j) \\
\mu_j &\sim N(95, 20) \\
\sigma_j &\sim \text{Inv-}\chi^2(10)
\end{aligned}
$$

Pooled model: in this model all the data from different machines are combined into a single model. This means that we treat all observations as if they came from a single group. In this case, there is a single μ and σ for all of the data.

$$
\begin{aligned}
y_i &\sim N(\mu, \sigma) \\
\mu &\sim N(95, 20) \\
\sigma &\sim \text{Inv-}\chi^2(10)
\end{aligned}
$$

Hierarchical model: this model allows for differences between groups but also assumes that there is some underlying structure that connects them, with shared parameters at the group level. In this case, each machine has its own μ_j and σ_j parameters, but all of them are drawn from a common distribution, therefore they all have common hyperparameters μ and τ.

Using Stan Prior Choice Recommendations

$$
\begin{aligned}
y_{ij} &\sim N(\mu_j, \sigma_j) \\
\mu_j &\sim N(\mu, \tau) \\
\mu &\sim N(95, 20) \\
\tau &\sim Half - t(4, 0, 20) \\
\sigma_j &\sim \text{Inv-}\chi^2(10)
\end{aligned}
$$

b) Implement the model in Stan and include the code in the report. Use weakly informative priors for all your models.

Separate model:

Here is the Stan code from `assignment7_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(100, 10);
    sigma[j] ~ inv_chi_square(10);
  }

  // likelihood
  for (j in 1:J)
    y[,j] ~ normal(mu[j], sigma[j]);
}

generated quantities {
  real ypred;
  // Compute predictive distribution
  ypred6 = normal_rng(mu[6], sigma[6]);
}
```

Here is the R implementation

```
options(mc.cores = parallel::detectCores())
sm_sep <- rstan::stan_model(file = "assignment7_separate.stan")

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

model_separate <- rstan::sampling(sm_sep, data = stan_data_separate)

df_summary_sep <- summarise_draws(model_separate)

df_summary_sep %>%  kbl(booktabs = T, linesep = "") %>%
  kable_styling(latex_options = c("striped", "scale_down", "HOLD_position"))
```

variable	mean	median	sd	mad	q5	q95	rhat	ess_bulk	ess_tail
mu[1]	77.864876	77.573911	6.248654	5.900437	68.260978	88.228080	1.0004736	4693.559	2442.746
mu[2]	105.810898	105.832216	3.861229	3.553763	99.424718	112.003606	1.0015122	5316.581	2902.399
mu[3]	88.007315	87.944985	4.240892	4.023549	81.200633	94.964880	1.0027622	4203.963	2593.387
mu[4]	111.366338	111.425120	2.522410	2.378045	107.247459	115.406535	1.0008942	4654.784	2293.187
mu[5]	90.057712	90.028220	3.640784	3.346987	84.255435	95.995368	1.0009941	4523.945	2194.288
mu[6]	86.831028	86.634268	6.204688	6.066816	76.954368	96.981878	1.0012707	4604.274	2968.762
sigma[1]	14.276848	13.575836	3.881490	3.231639	9.603370	21.506597	1.0021870	5132.893	2466.064
sigma[2]	8.583548	8.106935	2.374408	1.921932	5.773253	13.036465	1.0011852	4580.821	2410.815
sigma[3]	9.328363	8.943702	2.417486	2.118286	6.260906	13.724324	1.0024154	5290.221	3034.076
sigma[4]	5.526523	5.242245	1.476269	1.253393	3.683900	8.282986	1.0014382	5123.504	2661.273
sigma[5]	7.903843	7.484166	2.166739	1.808586	5.234608	11.958389	1.0013468	4987.365	2690.698
sigma[6]	14.073163	13.394094	3.854345	3.252138	9.408263	21.086601	1.0005761	4544.599	2492.753
ypred1	77.882660	77.647279	16.291878	15.364961	51.819265	104.454501	1.0008443	4261.727	3537.336
ypred2	105.653046	105.773043	9.655628	8.591221	89.907260	120.980578	1.0008722	4019.484	3612.075
ypred3	88.212913	88.117526	10.085815	9.347502	71.966373	105.597039	1.0005136	3976.190	3285.693
ypred4	111.363424	111.411620	6.383480	5.853716	100.982620	121.657170	0.9993720	4032.864	3836.181
ypred5	90.283557	90.320978	9.033556	8.373359	75.402292	104.643786	1.0005517	3572.062	3558.931
ypred6	86.756282	86.643403	15.810715	14.859960	60.889625	112.312110	0.9998099	4131.423	3690.143
lp___	-164.122211	-163.802516	2.597362	2.513085	-168.705301	-160.437619	1.0020855	1656.287	2190.021

Pooled model:

Here is the Stan code from `assignment7_pooled.stan`

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

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

model {
  // priors
  mu ~ normal(100, 10);
  sigma ~ inv_chi_square(10);

  // likelihood
  y ~ normal(mu, sigma);
}

generated quantities {
  real ypred;
  // Compute predictive distribution
  ypred = normal_rng(mu, sigma);
}
```

Here is the R implementation

```
options(mc.cores = parallel::detectCores())
sm_pool <- rstan::stan_model(file = "assignment7_pooled.stan")

stan_data_pooled <- list(
  y = unname(unlist(factory)),
  N = nrow(factory) * ncol(factory)
```

```
)

model_pooled <- rstan::sampling(sm_pool, data = stan_data_pooled)

df_summary_pool <- summarise_draws(model_pooled)

df_summary_pool %>%  kbl(booktabs = T, linesep = "") %>%
  kable_styling(latex_options = c("striped", "scale_down", "HOLD_position"))
```

variable	mean	median	sd	mad	q5	q95	rhat	ess_bulk	ess_tail
mu	92.90561	92.91770	3.014569	2.9279312	87.95728	97.8001	1.000818	3465.906	2531.852
sigma	16.98007	16.74055	2.129432	1.9760833	13.85237	20.8592	1.001227	3537.559	2574.574
ypred	92.74970	92.73965	17.414249	17.3971415	64.38383	121.8886	1.000396	4218.641	3971.852
lp___	-116.41611	-116.08411	1.036495	0.7014285	-118.51189	-115.4615	1.001374	1814.294	2226.371

Hierarchical model:

Here is the Stan code from `assignment7_h.stan`

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

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

model {
  alpha ~ normal(95, 20);
  tau ~ student_t(4, 0, 20);
  // priors
  for (j in 1:J){
    mu[j] ~ normal(alpha, tau);
    sigma[j] ~ inv_chi_square(10);
  }

  // likelihood
  for (j in 1:J)
    y[,j] ~ normal(mu[j], sigma[j]);
}

generated quantities {
  real ypred1;
  real ypred2;
  real ypred3;
  real ypred4;
  real ypred5;
  real ypred6;
  real ypred7;
```

```
  real mu_pred7;
  real sigma_pred7;
  // Compute predictive distribution
  ypred1 = normal_rng(mu[1], sigma[1]);
  ypred2 = normal_rng(mu[2], sigma[2]);
  ypred3 = normal_rng(mu[3], sigma[3]);
  ypred4 = normal_rng(mu[4], sigma[4]);
  ypred5 = normal_rng(mu[5], sigma[5]);
  ypred6 = normal_rng(mu[6], sigma[6]);
  mu_pred7 = normal_rng(alpha, tau);
  sigma_pred7 = inv_chi_square_rng(10);
  ypred7 = normal_rng(mu_pred7, sigma_pred7);
}
```

Here is the R implementation

```
options(mc.cores = parallel::detectCores())
sm_h <- rstan::stan_model(file = "assignment7_h.stan")

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

model_h <- rstan::sampling(sm_h, data = stan_data_h)

df_summary_h <- summarise_draws(model_h)

df_summary_h %>%  kbl(booktabs = T, linesep = "") %>%
  kable_styling(latex_options = c("striped", "scale_down", "HOLD_position"))
```

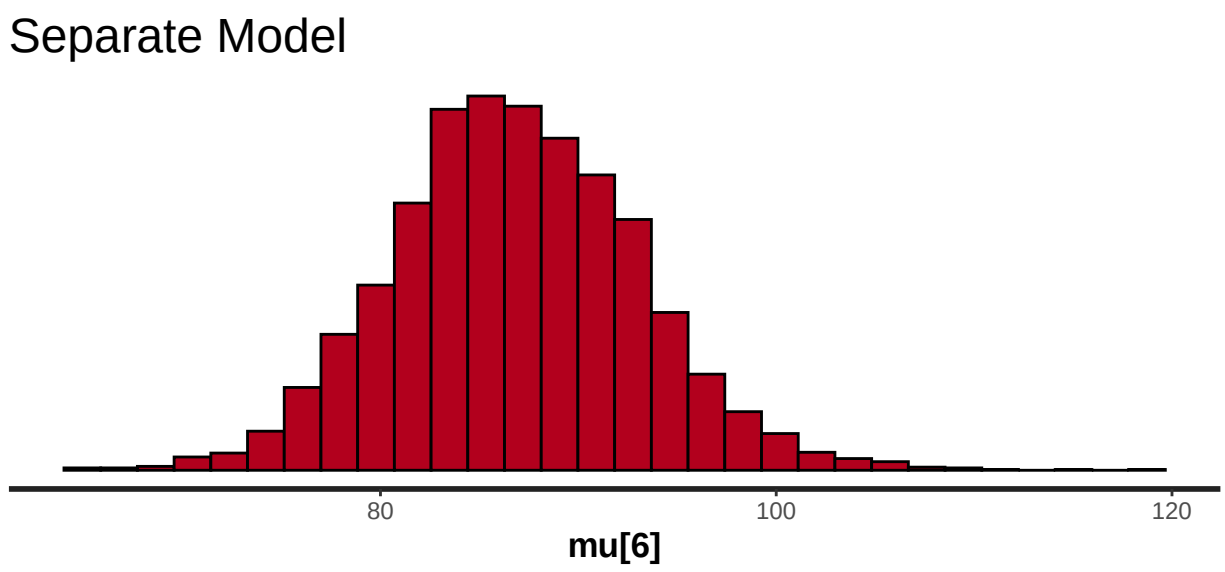
variable	mean	median	sd	mad	q5	q95	rhat	ess_bulk	ess_tail
mu[1]	79.9359419	79.5239902	6.5666867	6.1732680	69.922184	91.2697387	0.9999010	3063.267	2295.820
mu[2]	105.1079470	105.1842094	4.0458936	3.6210025	98.534603	111.5219126	1.0019775	3528.267	2319.135
mu[3]	88.3896319	88.3785896	3.9121899	3.6650849	82.042333	94.7982502	0.9999905	4487.198	2859.769
mu[4]	110.9137578	110.9835137	2.5471593	2.2631213	106.739194	114.8353399	1.0015348	3548.435	2414.622
mu[5]	90.2567543	90.2453244	3.4272351	3.1889132	84.690200	95.8796835	1.0031662	3684.231	2530.013
mu[6]	87.5058487	87.4329514	5.8443602	5.3337323	77.935080	97.4115357	1.0003805	3939.326	2649.477
sigma[1]	14.6387400	13.7718765	4.2245320	3.4182957	9.506747	22.6502491	1.0016161	3519.527	2543.378
sigma[2]	8.6801816	8.2943420	2.3428769	1.9254941	5.815355	12.9799387	1.0005575	4527.351	2391.428
sigma[3]	9.2599082	8.8271012	2.3713174	2.0525650	6.307869	13.7101661	0.9997342	4575.346	2828.903
sigma[4]	5.5517787	5.2667407	1.4955595	1.2418676	3.710174	8.3144995	1.0009952	3806.103	2417.278
sigma[5]	7.8658502	7.4865952	2.0847994	1.8791388	5.242898	11.8459922	1.0001820	4738.029	2967.585
sigma[6]	13.9912011	13.3598176	3.7510875	3.2493629	9.378458	20.5030428	1.0001685	4824.873	2516.040
mu_p	93.8793696	93.8992734	6.1113134	5.6851627	84.112422	103.2603280	1.0019609	3038.321	2628.683
tau	14.7657356	13.6268762	5.7754938	4.7551205	7.938708	25.3448618	1.0000310	3253.056	2605.857
ypred1	79.5928191	79.1447376	16.3127315	15.0956870	54.065119	106.5569407	0.9998202	3329.362	3495.932
ypred2	105.1884915	105.1875069	9.8804989	9.0539672	89.013770	120.9799733	1.0001182	3957.278	3687.291
ypred3	88.2401757	88.0595365	10.2959206	9.5265192	71.835581	105.0176016	1.0016405	3583.225	3372.282
ypred4	110.9055253	111.0506474	6.1126411	5.7685399	101.011641	120.9626096	1.0011781	3893.223	3316.834
ypred5	90.1867987	90.2857701	8.7043640	7.9480914	76.245493	104.1362229	1.0005397	3567.078	3559.112
ypred6	87.5297553	87.4598623	15.5887291	14.4561674	62.945168	113.3189594	1.0000172	4018.904	3651.470
ypred7	94.3356203	94.1197650	16.7371483	14.3972083	67.160089	121.2742884	1.0001090	4002.575	3968.258
mu_pred7	94.3342353	94.0991947	16.7386825	14.3841766	67.108089	121.2359223	1.0001053	4003.633	3927.050
sigma_pred7	0.1240344	0.1060589	0.0708745	0.0455572	0.054754	0.2582116	1.0001649	3985.256	3793.437
lp___	-179.3896168	-179.0579423	2.7970488	2.7218847	-184.422874	-175.4654200	1.0007291	1530.578	2355.212

c) Using the model (with weakly informative priors) report, comment on and, if applicable, plot histograms for the following distributions:

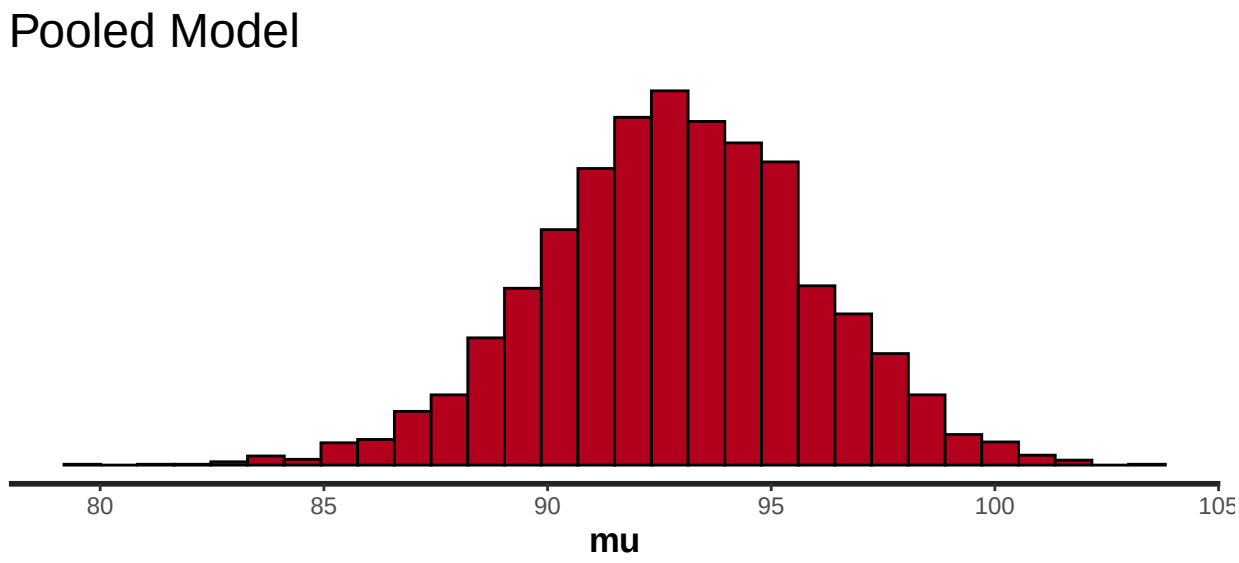
i) the posterior distribution of the mean of the quality measurements of the sixth machine.

In this case we can plot the histograms for all the posterior distributions of the mean of the quality measurements, note that in the case of the pooled model this histogram will always be the same no matter the machine, we are just plotting the pooled mean μ, because we are assuming they all belong in the same single group. The histograms in both the hierarchical and separate model look really similar, but if we see our point estimates we can see that in the hierarchical model we get a little higher mean and a little lower sd of the mean of the quality measurements of the sixth machine.

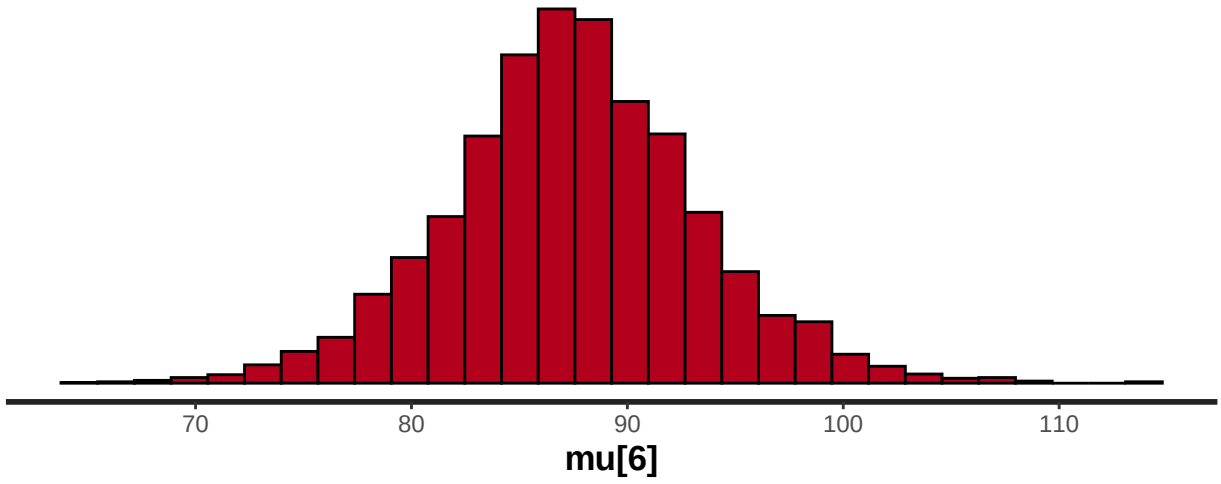
```
stan_hist(model_separate, pars = c('mu[6]')) + ggtitle("Separate Model")
```




```
stan_hist(model_pooled, pars = c('mu')) + ggtitle("Pooled Model")
```



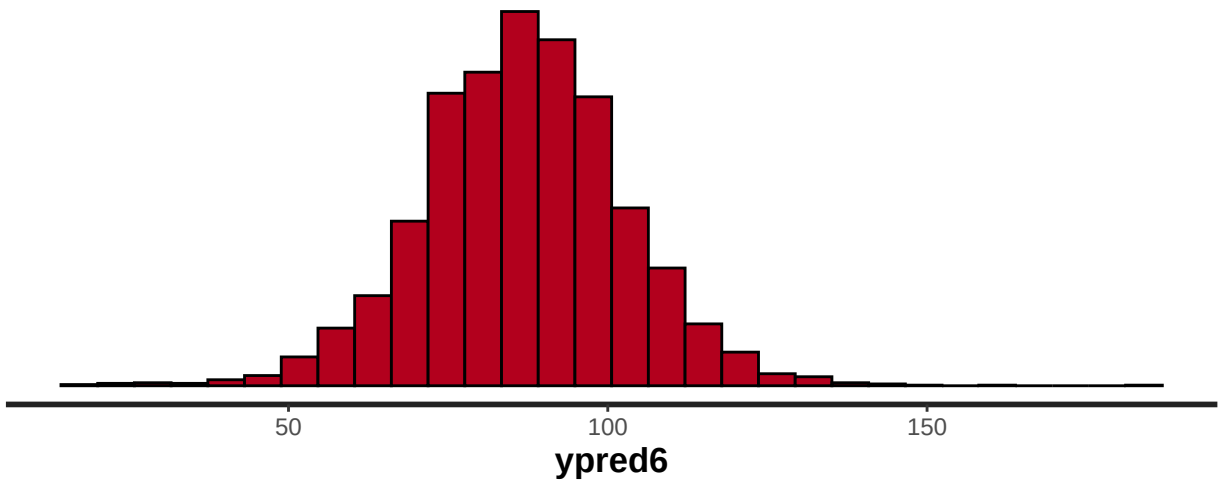

```
stan_hist(model_h, pars = c('mu[6]')) + ggtitle("Hierarchical Model")
```



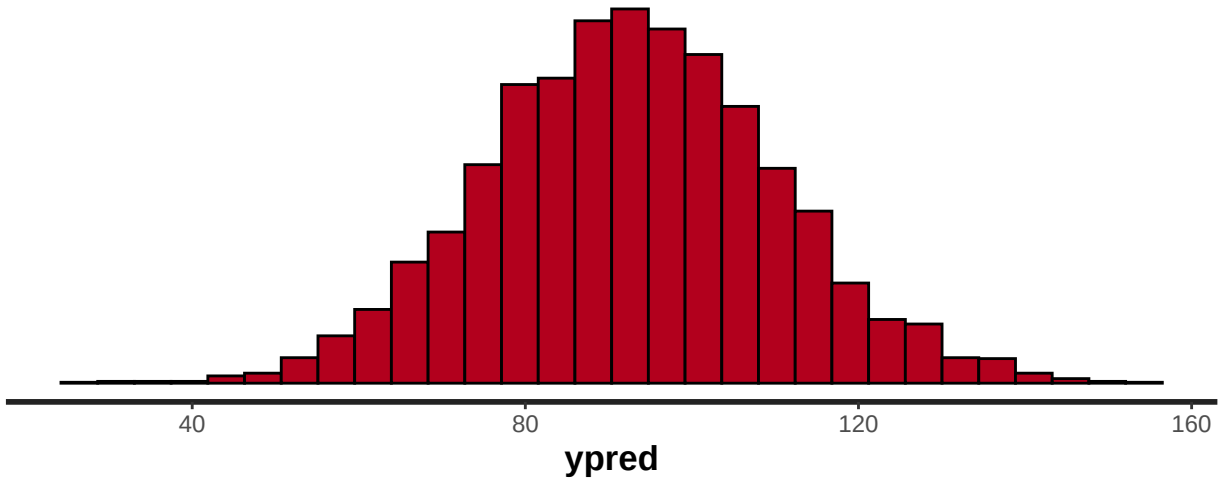
ii) the predictive distribution for another quality measurement of the sixth machine.

In this case we can also plot the histograms for all the predictive distributions for another quality measurement of the sixth machine, note that in the case of the pooled model this histogram is again always the same no matter the machine, we are just plotting the $\tilde{y}$. The histograms in both the hierarchical and separate model again look really similar, and our point estimates in both cases are also really close.

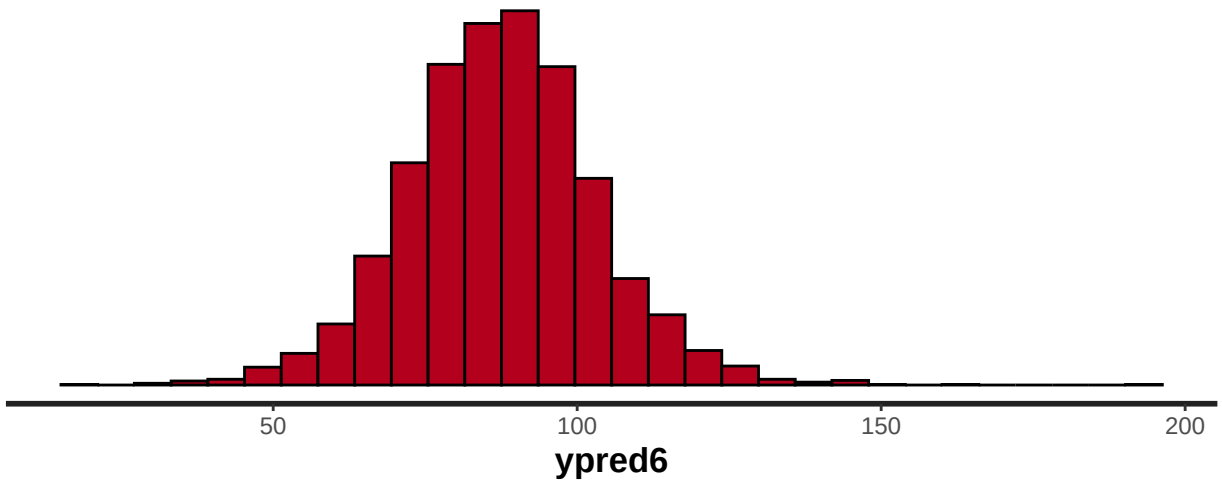
```
stan_hist(model_separate, pars = c('ypred6')) + ggtitle("Separate Model")
```




```
stan_hist(model_pooled, pars = c('ypred')) + ggtitle("Pooled Model")
```



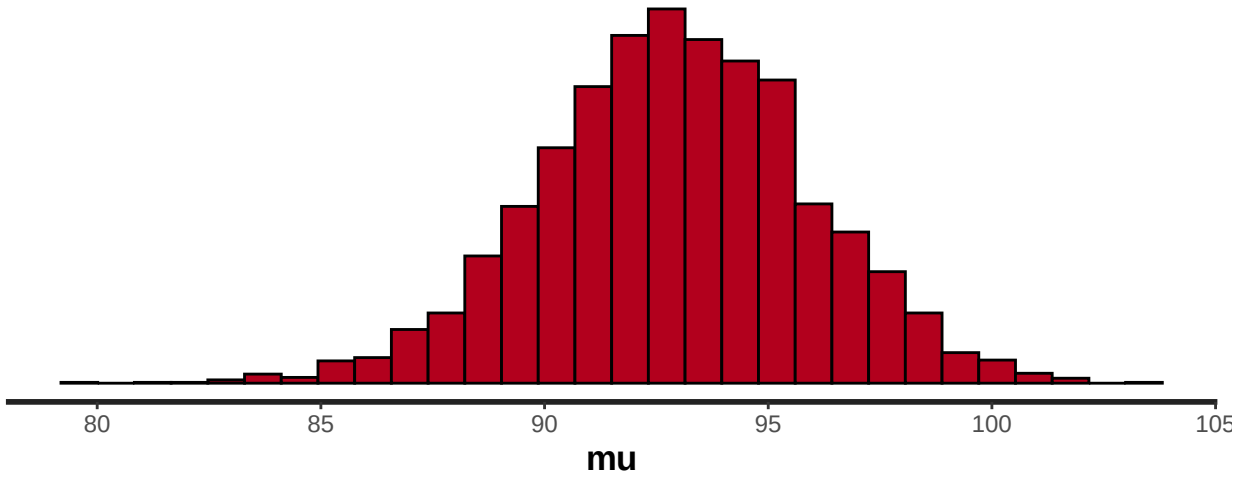

```
stan_hist(model_h, pars = c('ypred6')) + ggtitle("Hierarchical Model")
```



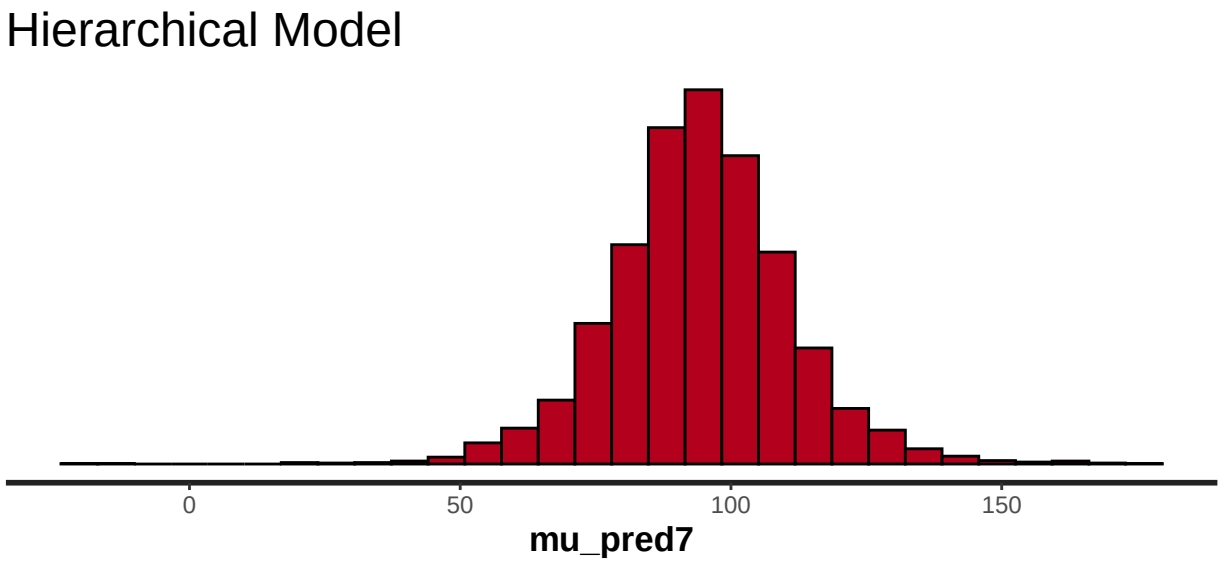
iii) the posterior distribution of the mean of the quality measurements of a seventh machine (not in the data).

Finally, In this case we cannot plot the histograms for all the posterior distribution of the mean of the quality measurements of a seventh machine, specifically we cannot do this for the separate model, since it considers each machine separately. On the contrary, in pooled models and hierarchical models we can get two types of predictive distributions: (i) A new observation in an existing group.(ii) A new observation in a new group. Note that in the case of the pooled model this histogram is the same as in question i),the pooled mean μ.

```
stan_hist(model_pooled, pars = c('mu')) + ggtitle("Pooled Model")
```




```
stan_hist(model_h, pars = c('mu_pred7')) + ggtitle("Hierarchical Model")
```



Report the posterior expectation for μ_1 with a 90% credible interval but using a $\text{normal}(0, 10)$ **prior for the** μ **parameter(s) and a** $\text{Gamma}(1, 1)$ **prior for the** σ **parameter(s). For the hierarchical model, use the** $\text{normal}(0, 10)$ **and** $\text{Gamma}(1, 1)$ **as hyper-priors.**

The code to get these results can be consulted in the Appendix at the end.

Table 1: Separate Model

variable	mean	mcse_mean	q10	q90	mcse_q10	mcse_q90
mu[1]	50.78889	0.1662487	38.84205	61.96437	0.3477361	0.3414156

Table 2: Pooled Model

variable	mean	mcse_mean	q10	q90	mcse_q10	mcse_q90
mu	85.58071	0.0838165	81.52708	89.52057	0.131306	0.0815115

Table 3: Hierarchical Model

variable	mean	mcse_mean	q10	q90	mcse_q10	mcse_q90
mu[1]	75.19574	0.0692219	69.22419	81.18475	0.0942753	0.1993136

We can report the the posterior expectation for μ_1 as 51 with a 90% credible interval of [39, 62] for the Separate model.

We can report the the posterior expectation for μ_1 as 85.6 with a 90% credible interval of [82, 89.5] for the Pooled model.

We can report the the posterior expectation for μ_1 as 75.2 with a 90% credible interval of [69, 81] for the Hierarchical model.

Appendix: Code for 2 d)

Separate model:

Here is the Stan code from `assignment7d_sep.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 ypred1;
  real ypred2;
  real ypred3;
  real ypred4;
  real ypred5;
  real ypred6;
  // Compute predictive distribution
  ypred1 = normal_rng(mu[1], sigma[1]);
  ypred2 = normal_rng(mu[2], sigma[2]);
  ypred3 = normal_rng(mu[3], sigma[3]);
  ypred4 = normal_rng(mu[4], sigma[4]);
  ypred5 = normal_rng(mu[5], sigma[5]);
  ypred6 = normal_rng(mu[6], sigma[6]);
}
```

Here is the R implementation

```
options(mc.cores = parallel::detectCores())
sm_d_sep <- rstan::stan_model(file = "assignment7d_sep.stan")

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

```r
model_d_separate <- rstan::sampling(sm_d_sep, data = stan_data_d_separate)

m_s <- extract_variable_matrix(model_d_separate, variable = 'mu[1]')
q_s <- mcse_quantile(m_s, probs = c(0.10, 0.90))

df_summary_d_s <- summarise_draws(model_d_separate,mean,mcse_mean,
                                  ~quantile(.x, probs = c(0.1, 0.9))) %>%
  filter(variable=='mu[1]')
df_summary_d_s$mcse_q10 <- q_s[1]
df_summary_d_s$mcse_q90 <- q_s[2]

df_summary_d_s %>%  kbl(booktabs = T, linesep = "", caption = "Separate Model") %>%
  kable_styling(latex_options = c("striped", "scale_down", "HOLD_position"))
```

Pooled model:

Here is the Stan code from `assignment7d_pool.stan`

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

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

model {
  // priors
  mu ~ normal(0, 10);
  sigma ~ gamma(1, 1);

  // likelihood
  y ~ normal(mu, sigma);
}

generated quantities {
  real ypred;
  // Compute predictive distribution
  ypred = normal_rng(mu, sigma);
}
```

Here is the R implementation

```r
options(mc.cores = parallel::detectCores())
sm_d_pool <- rstan::stan_model(file = "assignment7d_pool.stan")

stan_data_d_pooled <- list(
  y = unname(unlist(factory)),
  N = nrow(factory) * ncol(factory)
)

model_d_pooled <- rstan::sampling(sm_d_pool, data = stan_data_d_pooled)
```

```r
m_p <- extract_variable_matrix(model_d_pooled, variable = 'mu')
q_p <- mcse_quantile(m_p, probs = c(0.10, 0.90))

df_summary_d_p <- summarise_draws(model_d_pooled, mean,mcse_mean,
                                  ~quantile(.x, probs = c(0.1, 0.9))) %>%
  filter(variable=='mu')
df_summary_d_p$mcse_q10 <- q_p[1]
df_summary_d_p$mcse_q90 <- q_p[2]

df_summary_d_p %>%  kbl(booktabs = T, linesep = "", caption = "Pooled Model") %>%
  kable_styling(latex_options = c("striped", "scale_down", "HOLD_position"))
```

Hierarchical model:

Here is the Stan code from `assignment7d_h.stan`

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

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

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

  // likelihood
  for (j in 1:J)
    y[,j] ~ normal(mu[j], sigma[j]);
}

generated quantities {
  real ypred1;
  real ypred2;
  real ypred3;
  real ypred4;
  real ypred5;
  real ypred6;
  real ypred7;
  real mu_pred7;
  real sigma_pred7;
  // Compute predictive distribution
  ypred1 = normal_rng(mu[1], sigma[1]);
```

```
  ypred2 = normal_rng(mu[2], sigma[2]);
  ypred3 = normal_rng(mu[3], sigma[3]);
  ypred4 = normal_rng(mu[4], sigma[4]);
  ypred5 = normal_rng(mu[5], sigma[5]);
  ypred6 = normal_rng(mu[6], sigma[6]);
  mu_pred7 = normal_rng(mu_p, tau);
  sigma_pred7 = inv_chi_square_rng(10);
  ypred7 = normal_rng(mu_pred7, sigma_pred7);
}
```

Here is the R implementation

```
options(mc.cores = parallel::detectCores())
sm_d_h <- rstan::stan_model(file = "assignment7d_h.stan")

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

model_d_h <- rstan::sampling(sm_d_h, data = stan_data_d_h)

m_h <- extract_variable_matrix(model_d_h, variable = 'mu[1]')
q_h <- mcse_quantile(m_h, probs = c(0.10, 0.90))

df_summary_d_h <- summarise_draws(model_d_h,mean,mcse_mean,
                                  ~quantile(.x, probs = c(0.1, 0.9))) %>%
  filter(variable=='mu[1]')
df_summary_d_h$mcse_q10 <- q_h[1]
df_summary_d_h$mcse_q90 <- q_h[2]

df_summary_d_h %>%  kbl(booktabs = T, linesep = "", caption = "Hierarchical Model") %>%
  kable_styling(latex_options = c("striped", "scale_down", "HOLD_position"))
```