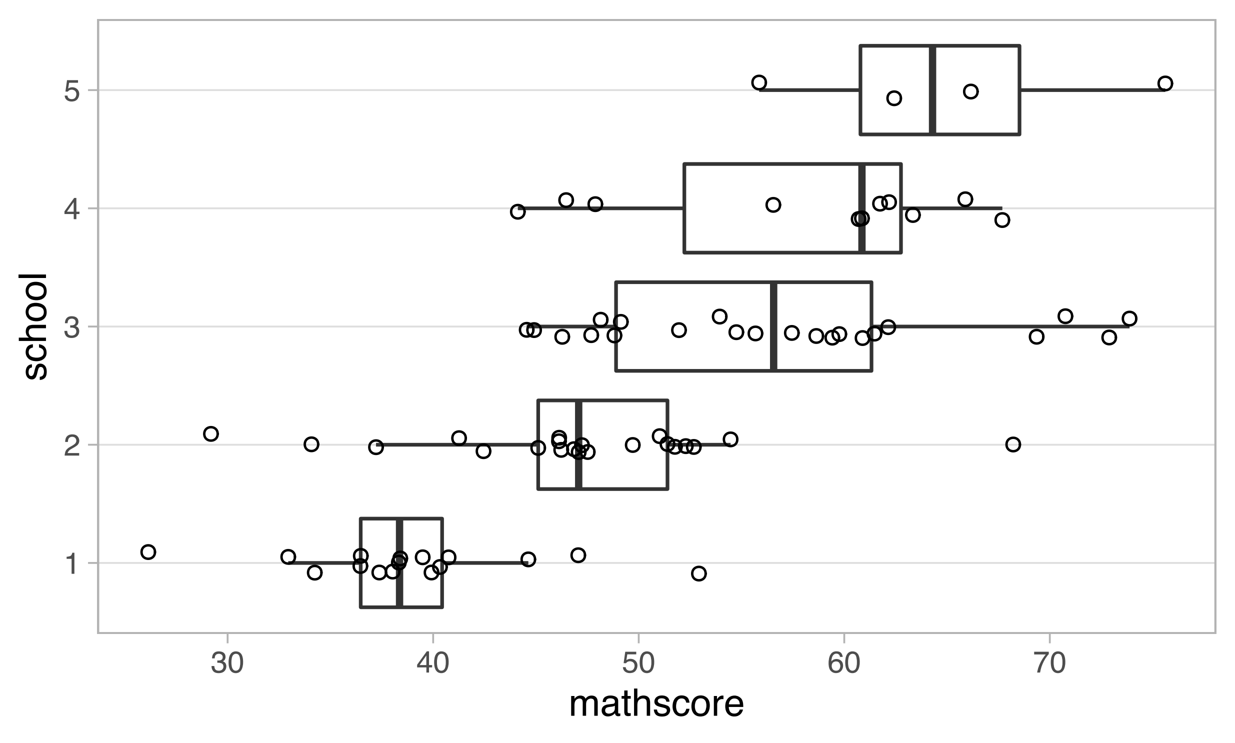
Hierarchical models

Stat 340: Bayesian Statistics

Example: ELS math scores

- 2002 Educational Longitudinal Study (ELS)
- Survey from schools across the United States
- Data are collected by sampling schools and then sampling students within each selected school
- We'll focus on 10th grade math scores from a sample of 10 schools
- Math tests contained items in arithmetic, algebra, geometry, data/probability, and advanced topics were divided into process categories of skill/knowledge, understanding/ comprehension, and problem solving

ELS math scores



Possible questions:

- What’s the typical math score?
- To what extent do scores vary from school to school?
- For any single school, how much might scores vary from student to student?

Possible analysis strategies

Complete pooling (combined estimates)
Ignore schools and lump all students together

No pooling (separate groups)
Separately analyze each school and assume that one school's data doesn't contain valuable information about another school

Partial pooling (compromise estimates)
Acknowledge the grouping structure, so that even though schools differ in performance, they might share valuable information about each other and about the broader population of schools

Your turn: Complete pooling

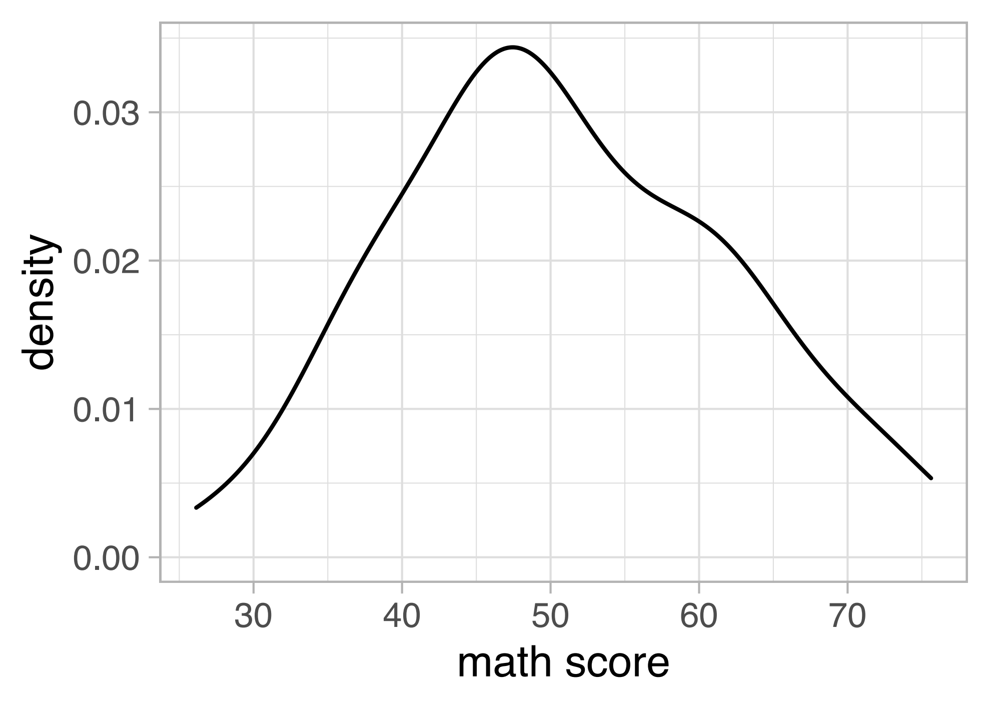
Suppose we decide to pool all of the scores together, disregarding what school each comes from.

Let Y_i denote the exam score for student i.

Write down a model (likelihood and priors) that could be used for the exams scores in this setting.

Just list distributions, don't do any algebra.

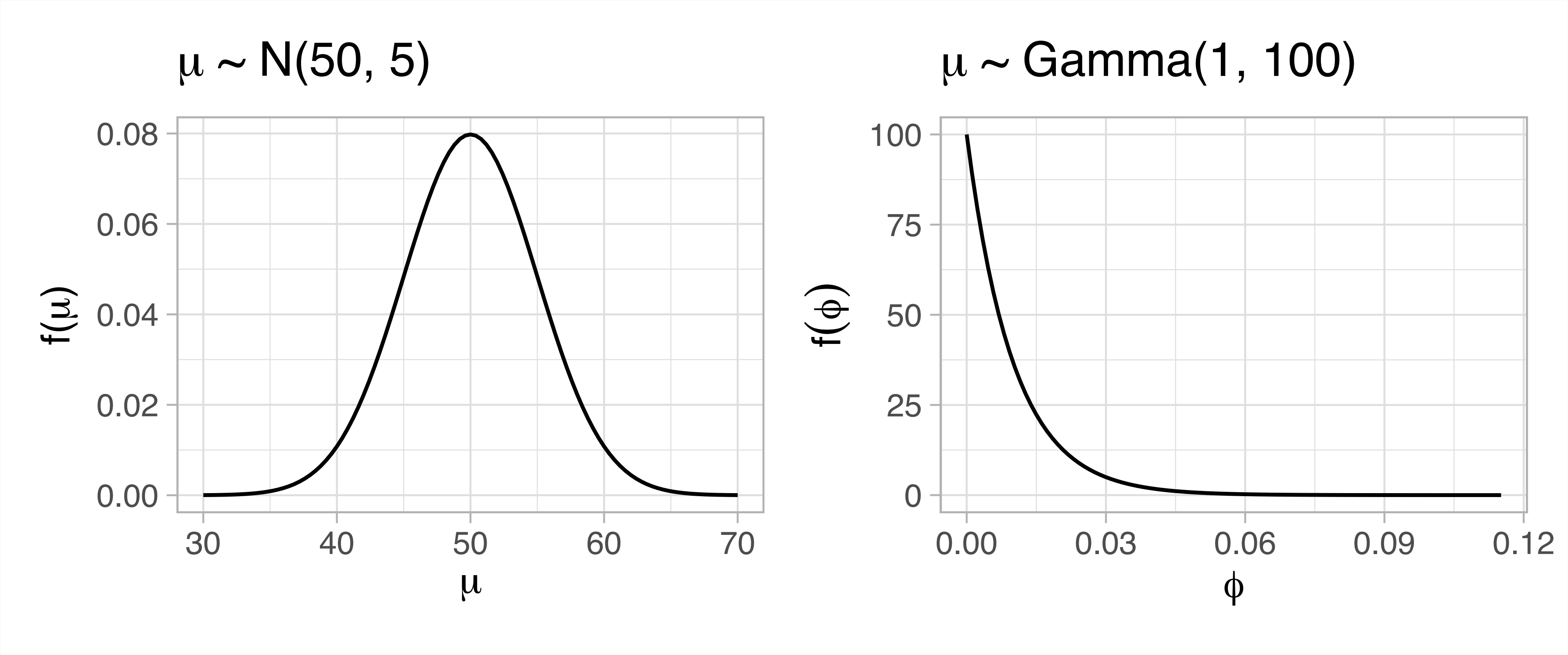
Pooling the scores together



- Density plot of the math scores shows the variability from score to score
- Range from 26 to 76
- Median: ~49

Prior knowledge

- Exam is standardized to have a mean of 50 and a standard deviation of 10
- Mean is most likely between 40 and 60 (let's call that a 95% interval)



Write the model for JAGS

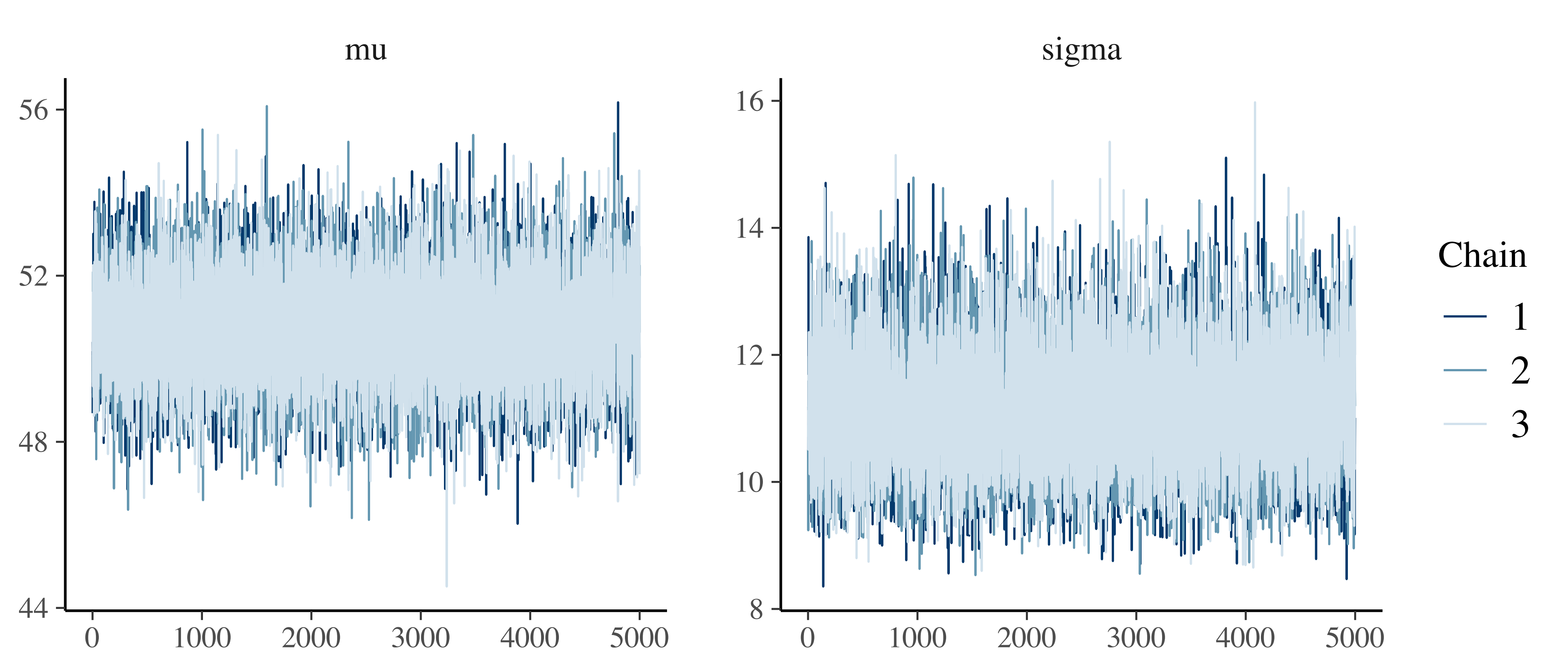
```
complete_pooling <-"
model {
## sampling
for (i in 1:N){
   y[i] ~ dnorm(mu, phi)
}

## priors
mu ~ dnorm(50, 1/25)  # remember to use precision here
phi ~ dgamma(1, 1/100)
sigma <- sqrt(pow(phi, -1))
}
"
```

Run MCMC

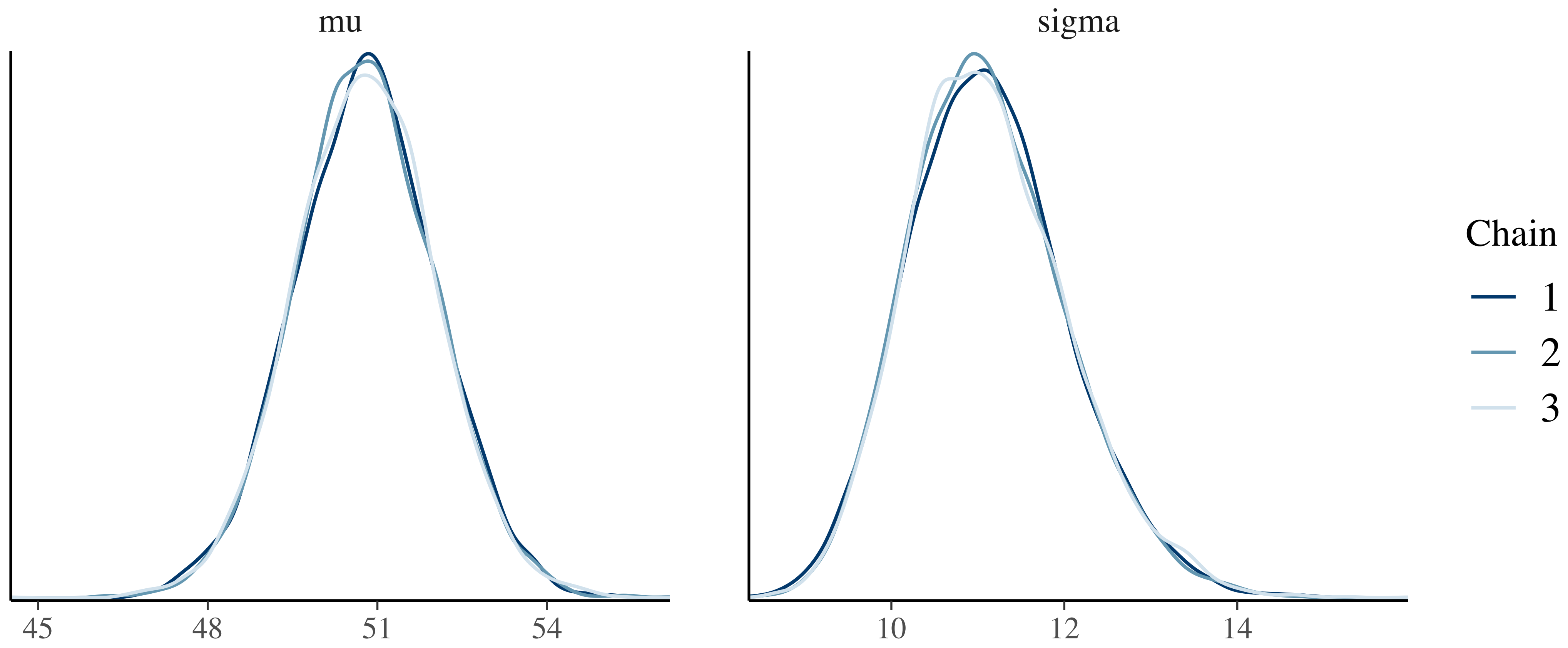
```
pooled_model <- run.jags(
  complete_pooling,
  data = list(y = sub_school$mathscore, N = length(sub_school$mathscore)),
  inits = list(list(mu = rnorm(1, 50, sd = 5), phi = runif(1, .1, .3)),
               list(mu = rnorm(1, 50, sd = 5), phi = runif(1, .1, .3)),
               list(mu = rnorm(1, 50, sd = 5), phi = runif(1, .1, .3))),
  monitor = c("mu", "sigma"),
  n.chains = 3,
  sample = 5000,
  silent.jags = TRUE
)
```

Remember to check for convergence before moving on to inference

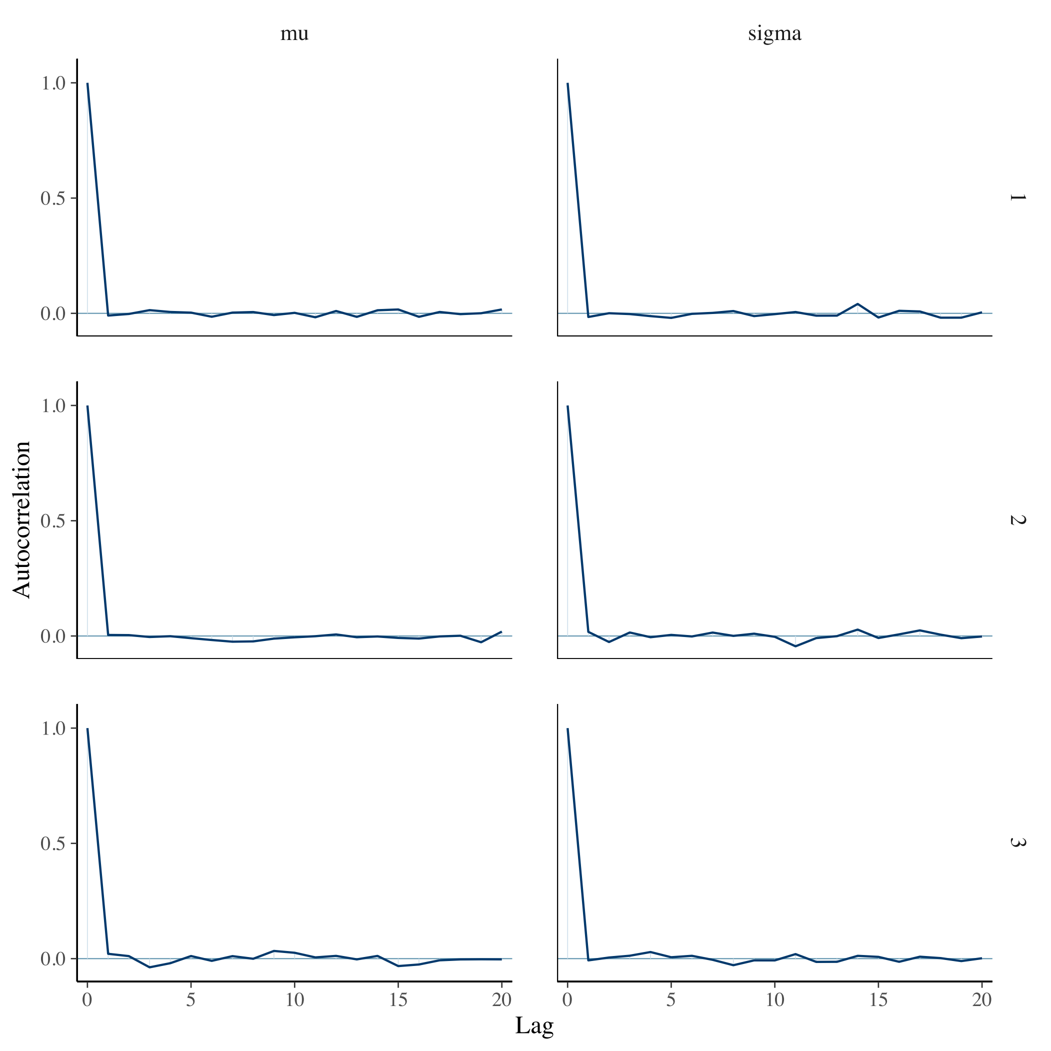
```
mcmc_trace(pooled_model$mcmc)
```



```
mcmc_dens_overlay(pooled_model$mcmc)
```



```
mcmc_acf(pooled_model$mcmc)
```



MCMC summary

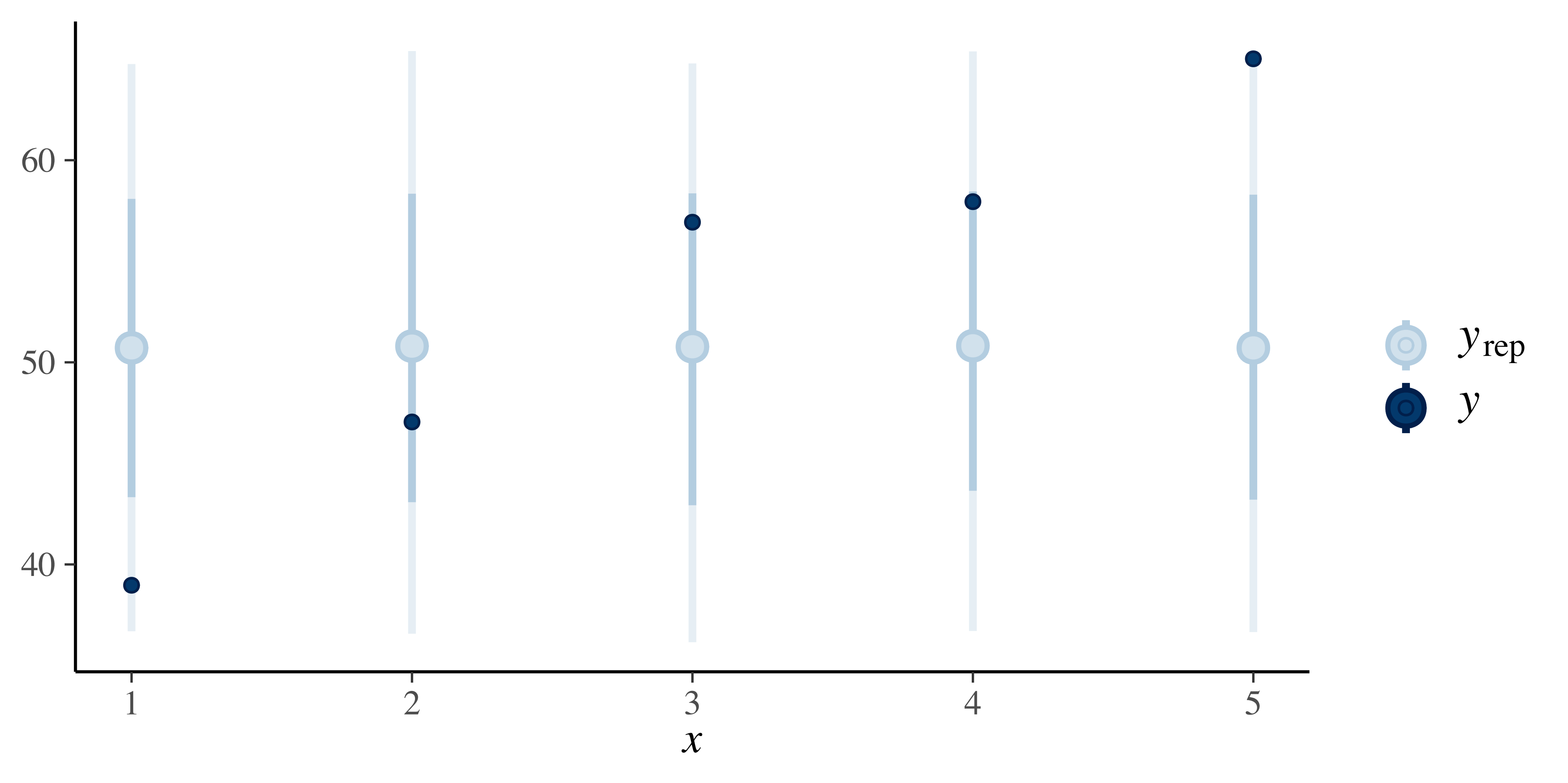
```r
print(pooled_model, digits = 3)
```

```
##
## JAGS model summary statistics from 15000 samples (chains = 3; adapt+burnin = 5000):
##
##       Lower95 Median Upper95 Mean    SD Mode   MCerr MC%ofSD SSeff     AC.10
## mu       48.3   50.8    53.2 50.8  1.25 50.8  0.0102     0.8 15064   0.00734
## sigma    9.42   11.1      13 11.1 0.917 10.9 0.00746     0.8 15085  -0.00486
##
##       psrf
## mu       1
## sigma    1
##
## Total time taken: 0.9 seconds
```

Typical score is around 50.8

Standard deviation from student to student is about 11.12

Pooled predictions vs. sample means



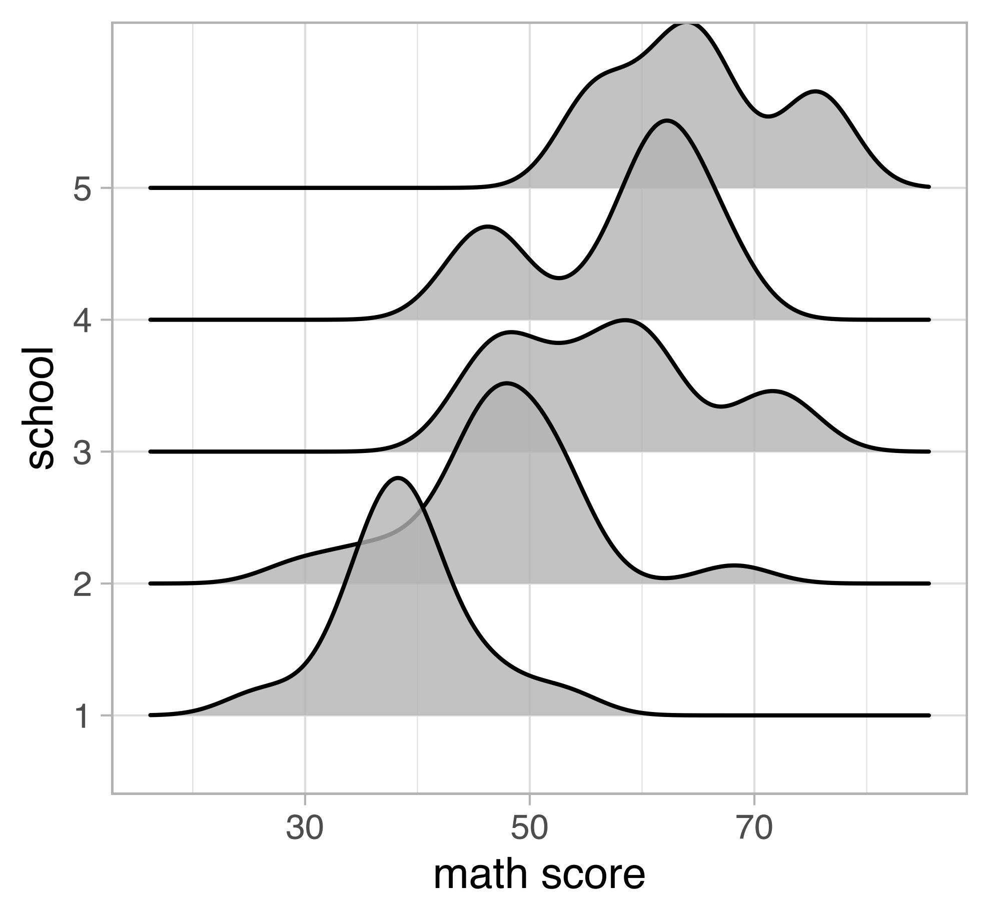
Your turn: No pooling

Suppose we decide to model each school's scores separately. In this case, we would need 10 models, one for each school.

Let Y_{ij} denote the exam score for student i in school j. (Book's notation)

Write down a model (likelihood and priors) that could be used for the exams scores in school j.

Separating the scores



- Typical values appear to vary by school
- Variability is similar between schools (remember, sample sizes aren't big, range from 4 to 22)
- Seems like we can simply from σ_j to σ in our model

Write the model for JAGS

```
no_pooling <- "
  model {
  ## Sampling
    for (i in 1:N){
      y[i] ~ dnorm(mu[school[i]], phi)
    }

    ## Priors
    for (j in 1:J){
      mu[j]  ~ dnorm(50, 1/100)
    }

    phi ~ dgamma(1, 1/100)
    sigma <- 1 / sqrt(phi)
  }
"
```

Run MCMC

```r
no_pooled_model <- run.jags(
  no_pooling,
  data = list(y = sub_school$mathscore, school = sub_school$school,
              N = nrow(sub_school), J = n_distinct(sub_school$school)),
  monitor = c("mu", "sigma"),
  n.chains = 3,
  sample = 5000,
  silent.jags = TRUE
)
```

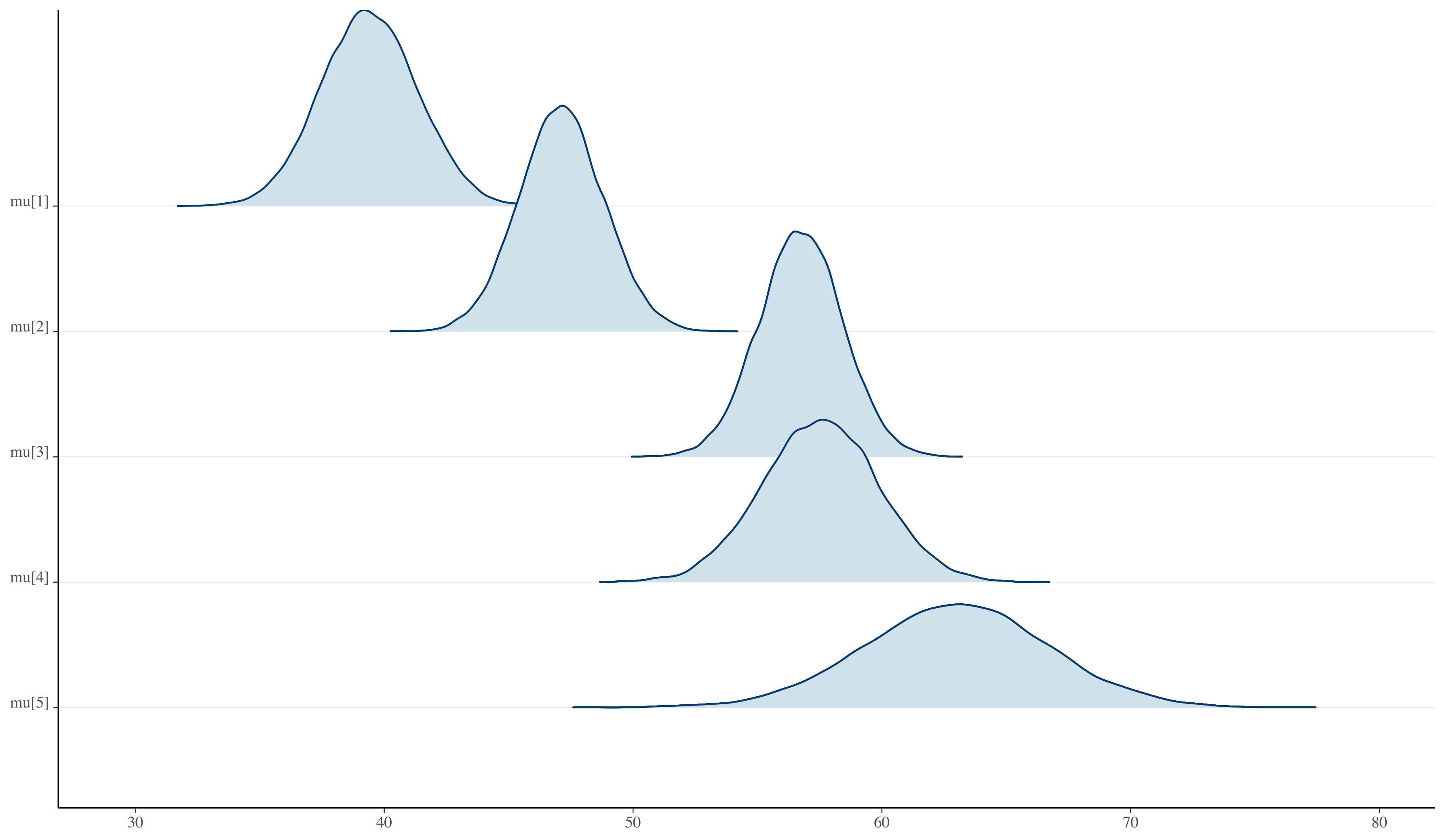
```r
mcmc_trace(no_pooled_model$mcmc,  regex_pars = "mu")
```

MCMC summary

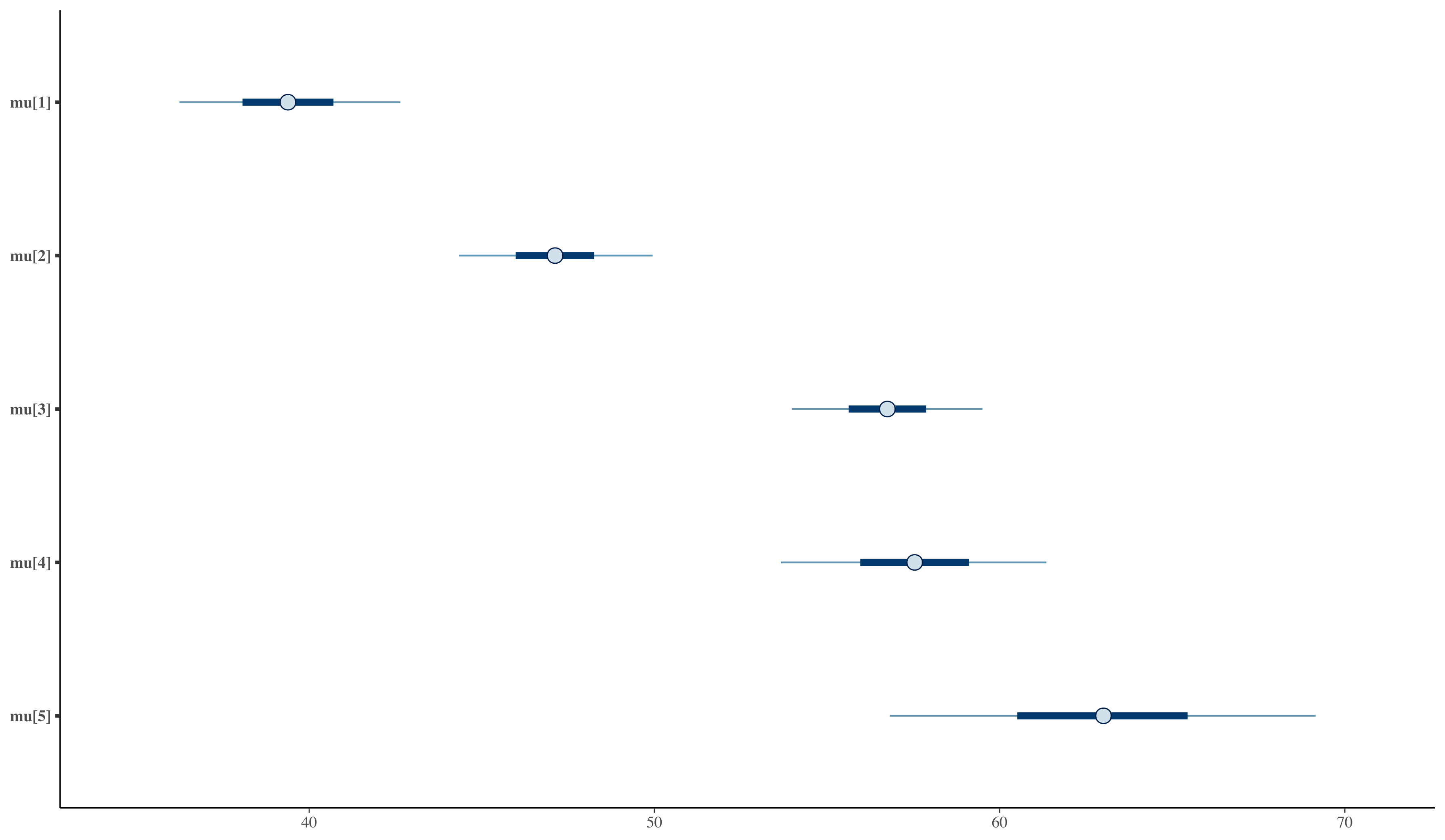
```r
print(no_pooled_model, digits = 3)
```

```
##
## JAGS model summary statistics from 15000 samples (chains = 3; adapt+burnin = 5000):
##
##       Lower95 Median Upper95 Mean    SD Mode   MCerr MC%ofSD SSeff    AC.10
## mu[1]    35.5   39.4    43.1 39.4  1.95 39.4  0.0158     0.8 15227 -0.00559
## mu[2]    43.8   47.1    50.5 47.1  1.71 47.1  0.0139     0.8 15000  0.00403
## mu[3]    53.4   56.7      60 56.7  1.68 56.8  0.0136     0.8 15388 -0.00383
## mu[4]    53.1   57.5    62.3 57.5  2.34 57.7  0.0194     0.8 14539 0.000529
## mu[5]    55.6     63    70.2   63  3.73 63.1  0.0307     0.8 14730  0.00487
## sigma    6.69    7.9     9.3 7.95 0.676 7.85 0.00593     0.9 13035 -0.00915
##
##       psrf
## mu[1]    1
## mu[2]    1
## mu[3]    1
## mu[4]    1
## mu[5]    1
## sigma    1
##
## Total time taken: 1.1 seconds
```

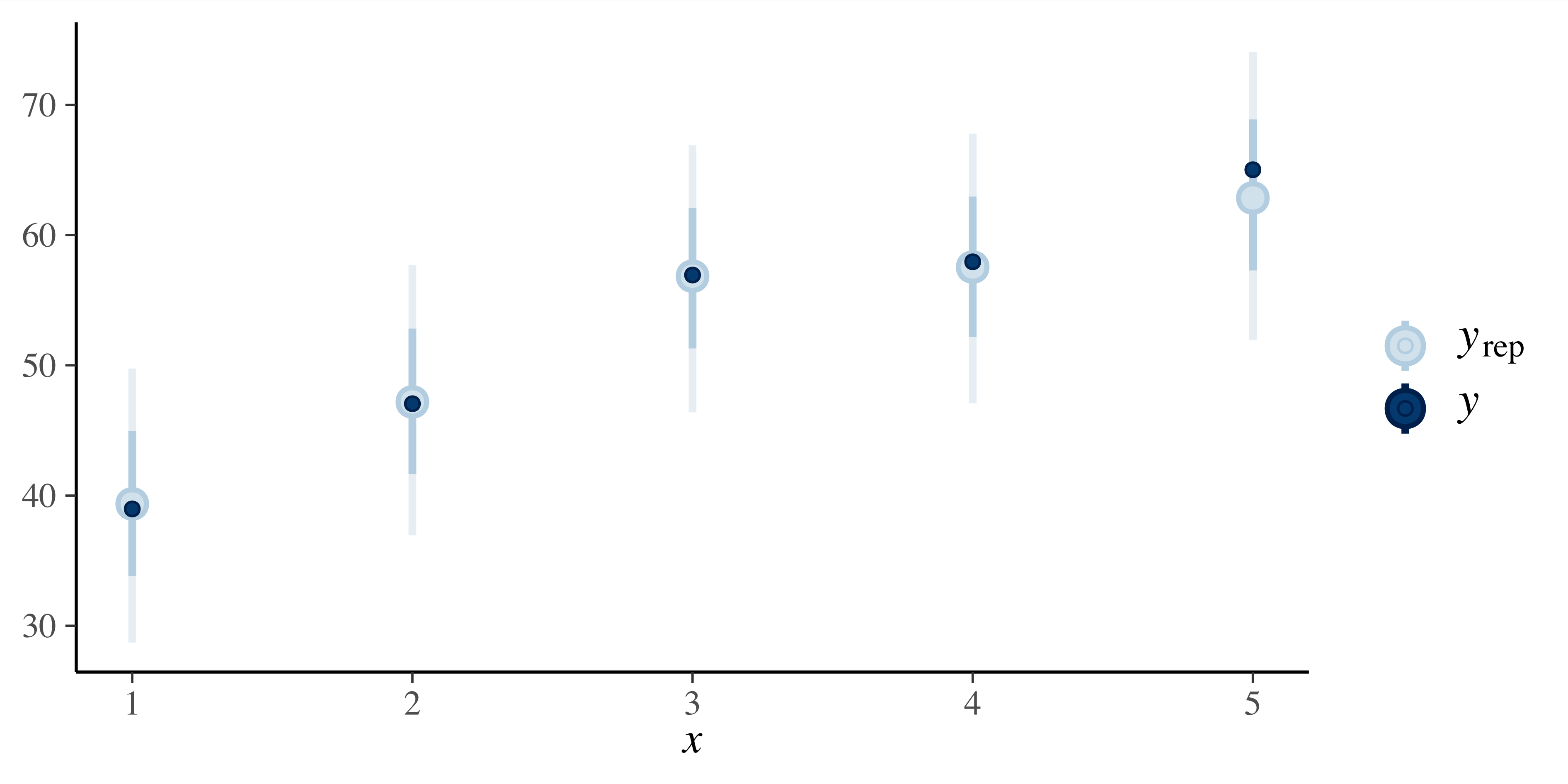
```
mcmc_areas_ridges(no_pooled_model$mcmc, regex_pars = "mu")
```



```r
mcmc_intervals(no_pooled_model$mcmc, regex_pars = "mu")
```



No pooled predictions vs. sample means



What have we seen so far?

- Completely pooled model does not acknowledge differences between schools
- No pooled model acknowledges that some schools tend to score higher than others
- No pooled model ignores data on one school when learning about the typical score of another
- No pooled model cannot be generalized to schools outside our sample

Hierarchical model

Let's compromise between the the complete pooled and no pooled models

How? By using a *two-stage prior* specification

Hierarchical model specification for JAGS

```
modelString <-"
model {

## sampling
for (i in 1:N){
   y[i] ~ dnorm(mu_j[school[i]], invsigma2)
}

## priors
for (j in 1:J){
   mu_j[j] ~ dnorm(mu, invtau2)
}

invsigma2 ~ dgamma(a_s, b_s)
sigma <- sqrt(pow(invsigma2, -1))

## hyperpriors
mu ~ dnorm(mu0, g0)
invtau2 ~ dgamma(a_t, b_t)
tau <- sqrt(pow(invtau2, -1))
}
"
```

Define the data and prior parameters

```
y <- sub_school$mathscore
school <- sub_school$school
N <- length(y)
J <- length(unique(school))
the_data <- list(y = y, school = school,
                 N = N, J = J,
                 mu0 = 50, g0 = .04,  # prior parameters
                 a_t = 1, b_t = .01,  # hyperparameters
                 a_s = 1, b_s = .01)  # hyperparameters
```

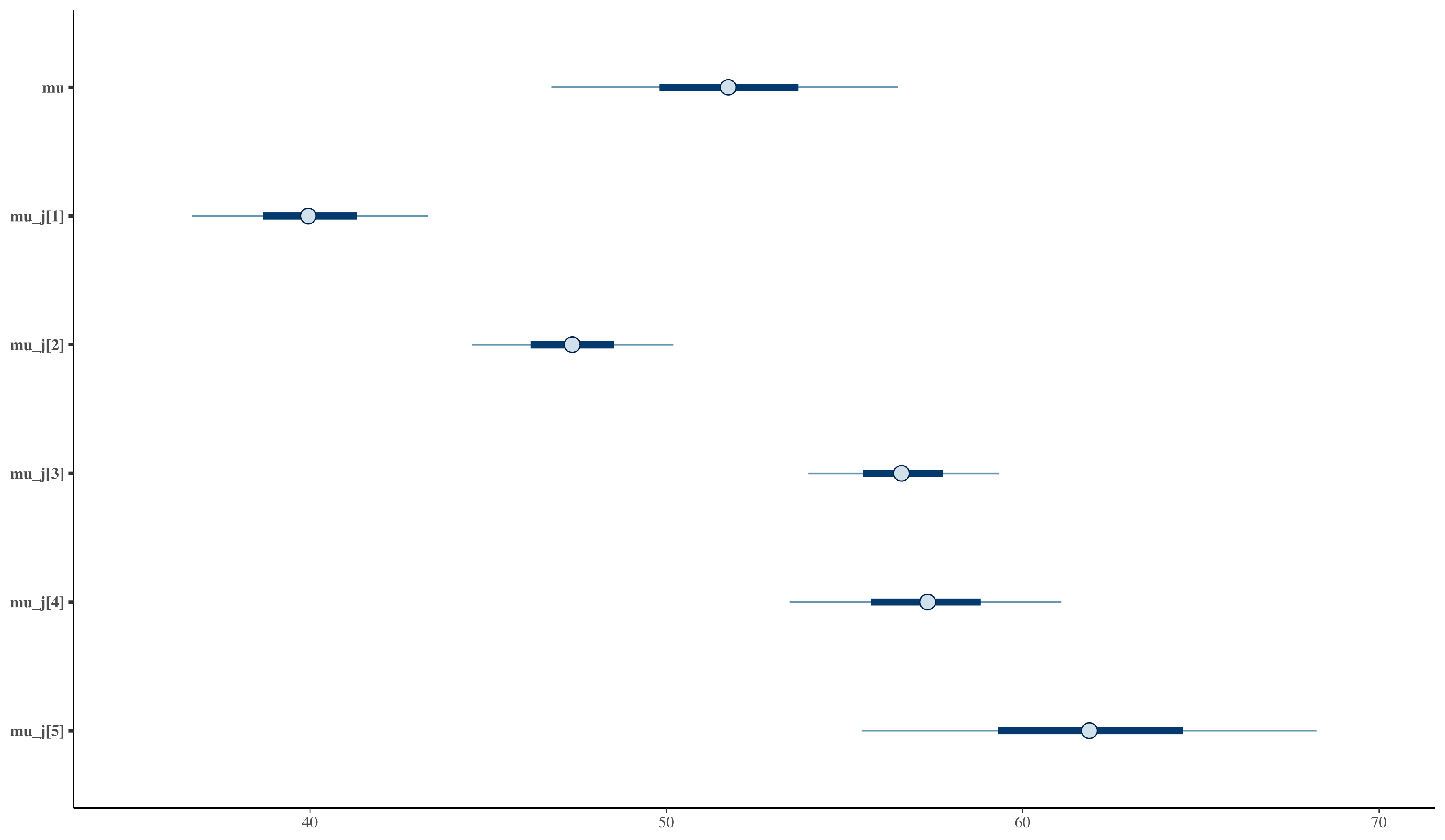
Run MCMC

```r
posterior <- run.jags(
  modelString,
  n.chains = 1,
  data = the_data,
  monitor = c("mu", "tau", "mu_j", "sigma"),
  adapt = 1000,
  burnin = 5000,
  sample = 5000,
  silent.jags = TRUE
)
```

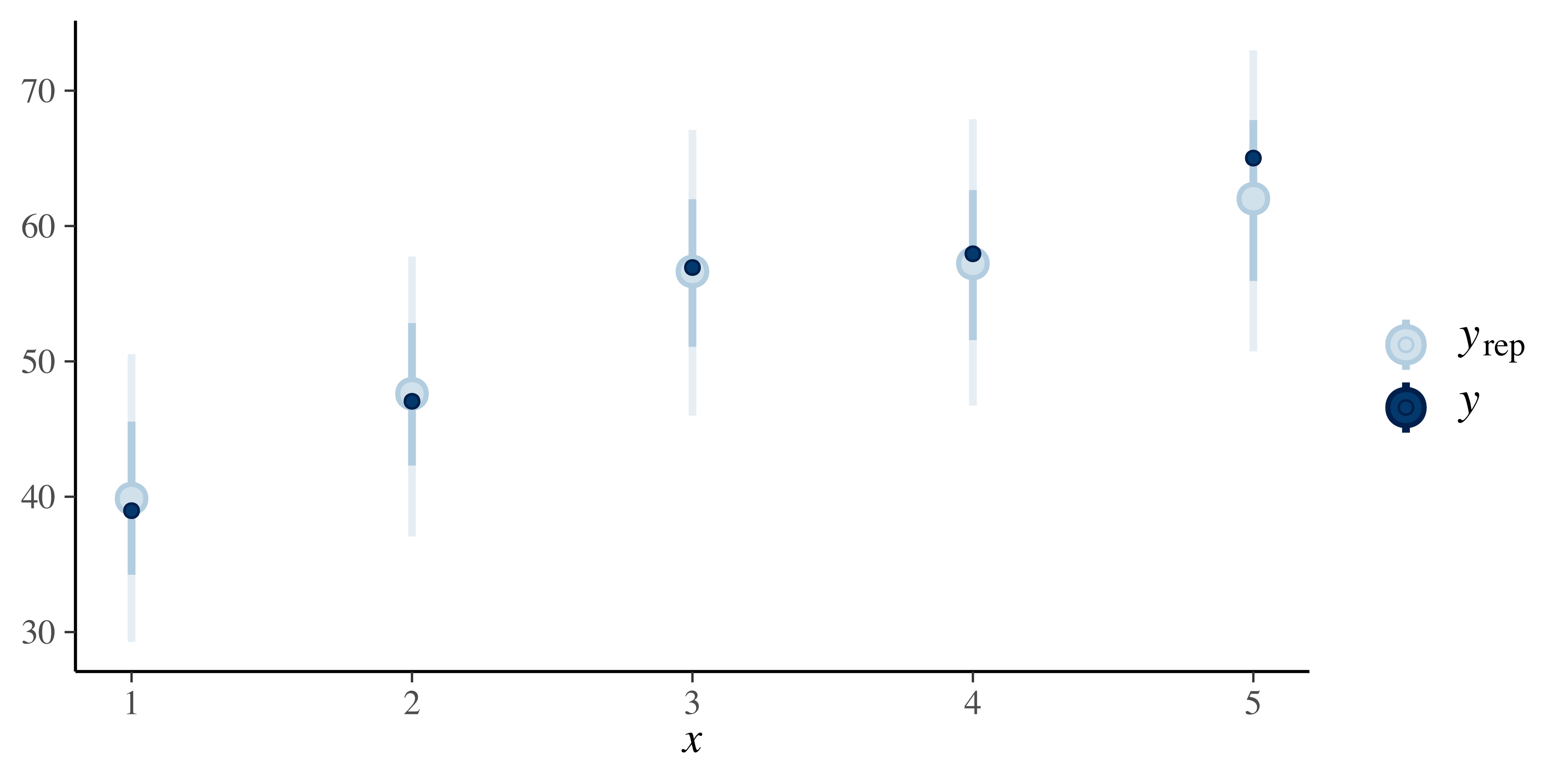
```r
print(posterior, digits = 3)
```

```
##
## JAGS model summary statistics from 5000 samples (adapt+burnin = 6000):
##
##          Lower95 Median Upper95 Mean    SD Mode  MCerr MC%ofSD SSeff     AC.10
## mu         45.7   51.7    57.5 51.7     3 51.8 0.0444     1.5  4557    -0.023
## tau        3.64   7.58    14.7 8.24  3.16 6.89  0.055     1.7  3288 -0.000269
## mu_j[1]    35.9     40    43.8   40  2.02   40 0.0323     1.6  3928    0.0217
## mu_j[2]    44.1   47.4    50.8 47.4  1.73 47.3 0.0254     1.5  4637   -0.0133
## mu_j[3]    53.3   56.6    59.8 56.6  1.66 56.7 0.0247     1.5  4516   -0.0137
## mu_j[4]    52.7   57.3    61.7 57.3  2.32 57.4  0.035     1.5  4414     0.021
## mu_j[5]    54.3   61.9    69.3 61.9  3.86 61.8 0.0676     1.8  3258  -0.00552
## sigma       6.7    7.9    9.36 7.97 0.687 7.78 0.0108     1.6  4078 -0.000919
##
##          psrf
## mu        --
## tau       --
## mu_j[1]   --
## mu_j[2]   --
## mu_j[3]   --
## mu_j[4]   --
## mu_j[5]   --
## sigma     --
##
## Total time taken: 0.2 seconds
```

```
mcmc_intervals(posterior$mcmc, regex_pars = "mu")
```



Hierarchical predictions vs. sample means


x
y_{rep}
y