

Slide 17 - R Output

Elvan Ceyhan

11/25/2024

```
#Set Working Directory to Source File Location  
library("rstudioapi") # Load rstudioapi package  
#setwd(dirname(getActiveDocumentContext()$path)) # Set working directory to source file location  
#getwd()
```

Some required packages:

```
# Load packages  
library(rstan)  
library(ggplot2)  
library(bayesrules)  
library(tidyverse)  
library(rstanarm)  
library(bayesplot)  
library(tidybayes)  
library(broom.mixed)
```

Chapter 17 R Examples from Bayes Rules! Book

```
# Load the data  
data(cherry_blossom_sample)  
running <- cherry_blossom_sample  
  
# Data preprocessing: Select relevant variables and remove missing values  
running <- running %>%  
  select(runner, age, net) %>%  
  drop_na() # Replaced na.omit() with drop_na() for tidyverse consistency
```

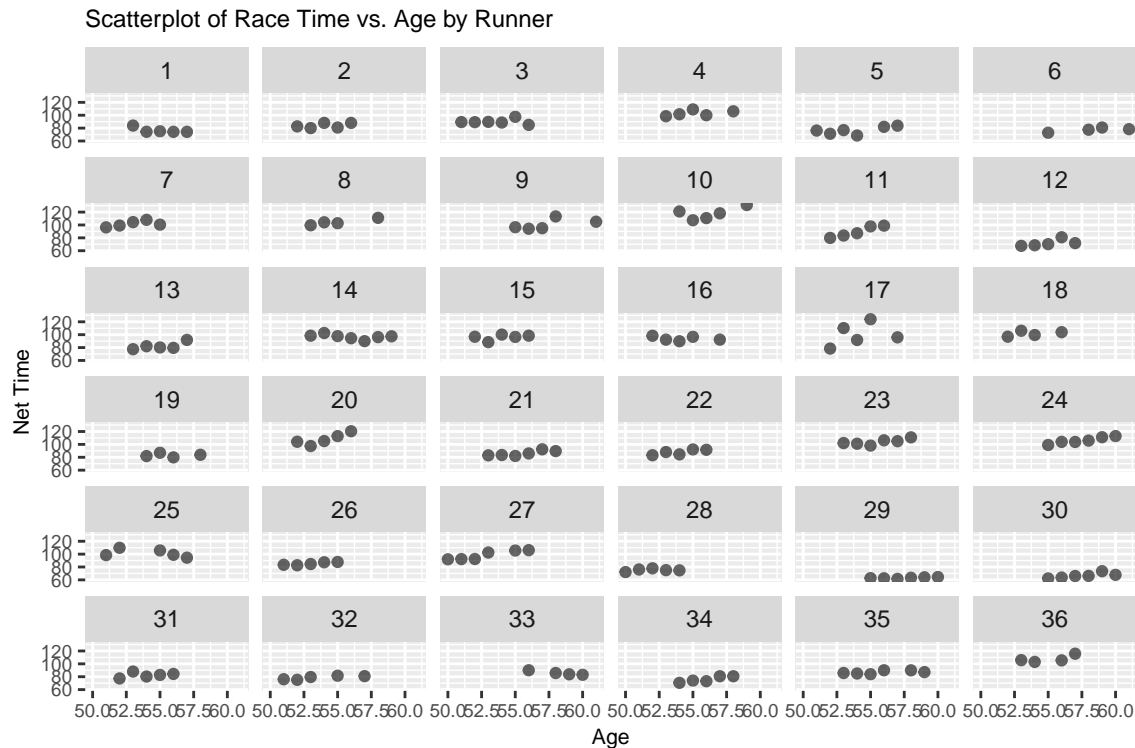
Fitting the hierarchical varying-intercepts model

This model allows the intercept to vary by runner, reflecting individual differences Note: The fitting process may be slow due to the large number of iterations

```
running_model_1 <- stan_glmer(  
  net ~ age + (1 | runner),  
  data = running, family = gaussian,  
  prior_intercept = normal(100, 10), # Prior for the global intercept  
  prior = normal(2.5, 1), # Prior for the slope (age effect)  
  prior_aux = exponential(1, autoscale = TRUE), # Prior for residual standard deviation  
  prior_covariance = decov(reg = 1, conc = 1, shape = 1, scale = 1), # Prior for group-level covariance  
  chains = 4, iter = 2000 # Reduced to 2000 iterations from 5000 for faster computation  
)
```

Visualize scatterplots of race time vs. age for each runner

```
# This helps to understand the variation across runners
ggplot(running, aes(x = age, y = net)) +
  geom_point(alpha = 0.6) + # Slight transparency for better visibility
  facet_wrap(~ runner) +
  labs(title = "Scatterplot of Race Time vs. Age by Runner",
       x = "Age", y = "Net Time")
```



Lots of parameters, may be best to examine these individually as in Chapter 16...

Summarizing posterior distributions for global parameters

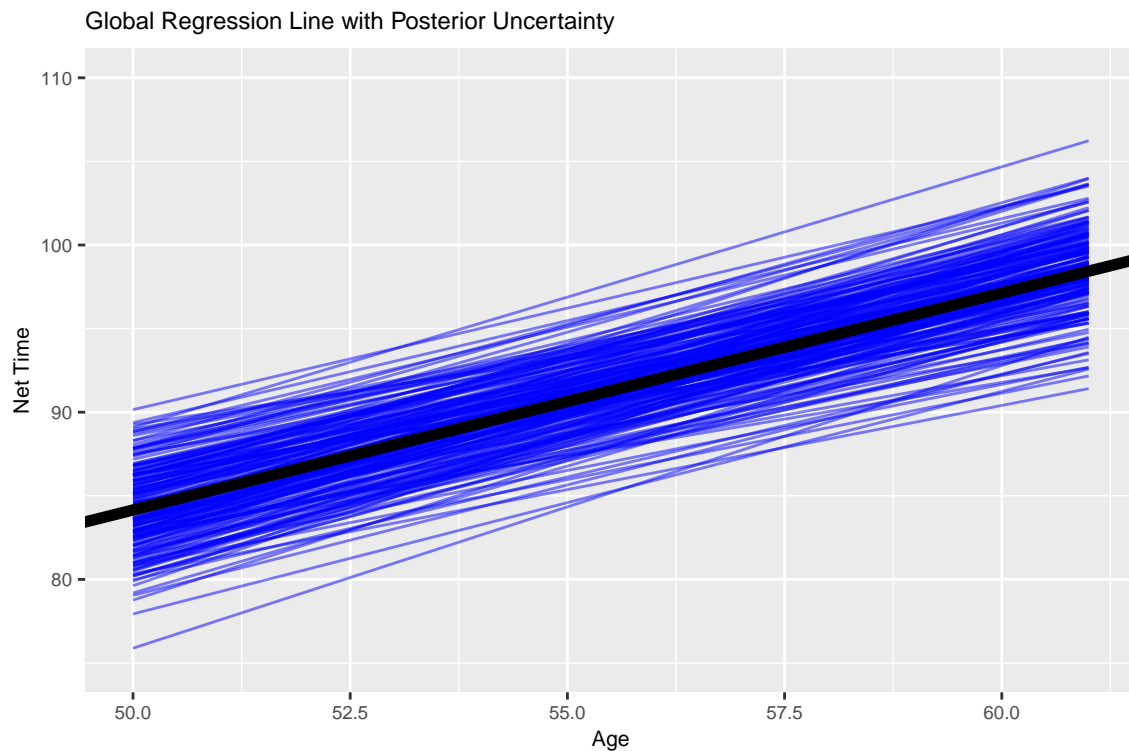
```
#Don't run too much clutter!
#mcmc_trace(running_model_1)
#mcmc_dens_overlay(running_model_1)
#mcmc_acf(running_model_1)

# Provides estimates of the global intercept and slope
tidy_summary_1 <- tidy(running_model_1, effects = "fixed",
                      conf.int = TRUE, conf.level = 0.80)
print(tidy_summary_1)
```

```
## # A tibble: 2 x 5
##   term      estimate std.error conf.low conf.high
##   <chr>      <dbl>    <dbl>   <dbl>   <dbl>
## 1 (Intercept) 19.2      11.8     3.48    34.8
## 2 age         1.30      0.214    1.02     1.58
```

Visualizing the global regression line with posterior uncertainty

```
# Adding posterior draws to show variability
B0 <- tidy_summary_1$estimate[1]
B1 <- tidy_summary_1$estimate[2]
running %>%
  add_fitted_draws(running_model_1, n = 200, re_formula = NA) %>%
  ggplot(aes(x = age, y = net)) +
  geom_line(aes(y = .value, group = .draw), alpha = 0.5, color = "blue") +
  geom_abline(intercept = B0, slope = B1, color = "black", linewidth=2) +
  lims(y = c(75, 110)) +
  labs(title = "Global Regression Line with Posterior Uncertainty",
       x = "Age", y = "Net Time")
```



Posterior summaries of runner-specific intercepts

Shows individual-level adjustments to the intercept

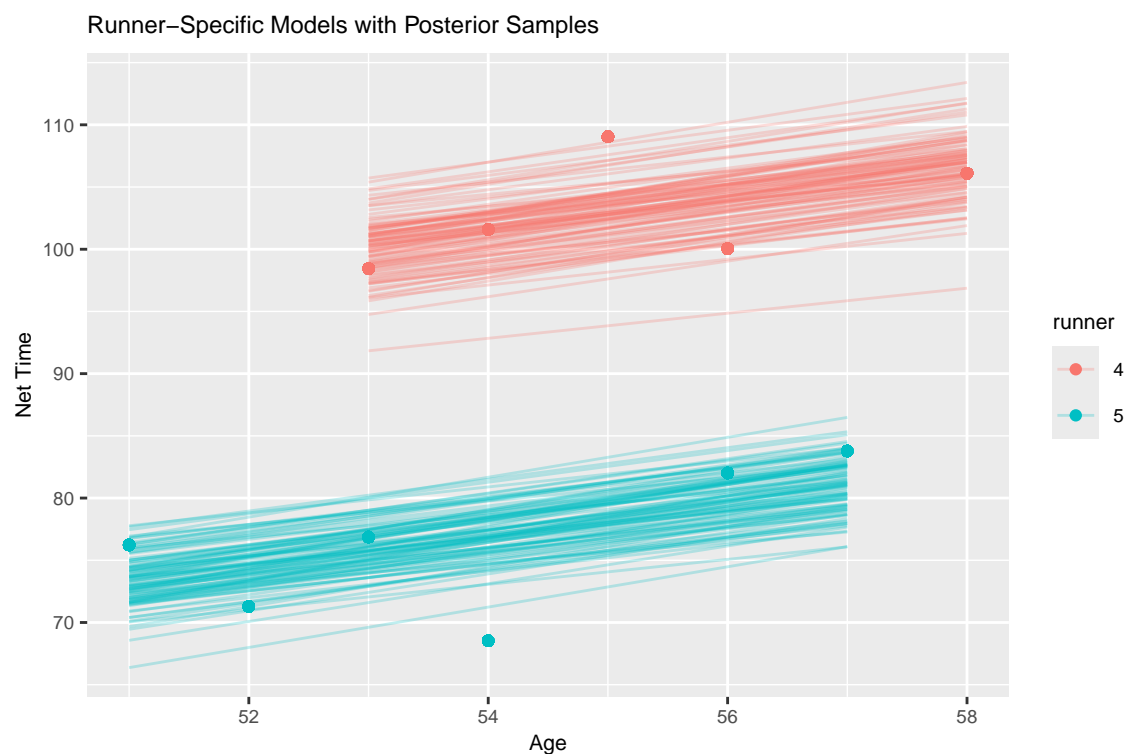
```
runner_summaries_1 <- running_model_1 %>%
  spread_draws(`(Intercept)`, b[,runner]) %>%
  mutate(runner_intercept = `(Intercept)` + b) %>%
  select(`(Intercept)`, -b) %>%
  median_qi(.width = 0.80) %>%
  select(runner, runner_intercept, .lower, .upper)

# Filtering summaries for specific runners (4 and 5)
runner_summaries_1 %>%
  filter(runner %in% c("runner:4", "runner:5"))
```

```
## # A tibble: 2 x 4
```

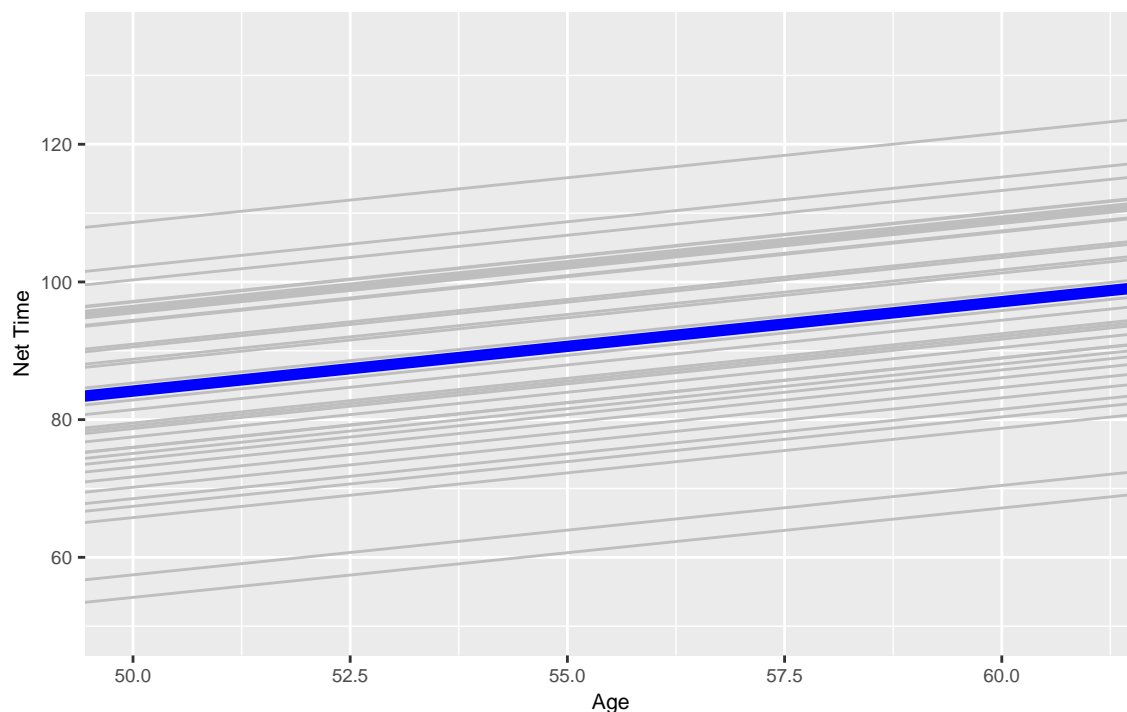
```
## runner runner_intercept .lower .upper
## <chr> <dbl> <dbl> <dbl>
## 1 runner:4 30.8 15.3 46.4
## 2 runner:5 6.74 -8.48 21.7

# Posterior plausible models for runners 4 and 5
# Adds fitted draws for visualization
running %>%
  filter(runner %in% c("4", "5")) %>%
  add_fitted_draws(running_model_1, n = 100) %>%
  ggplot(aes(x = age, y = net)) +
  geom_line(aes(y = .value, group = paste(runner, .draw), color = runner), alpha = 0.25) +
  geom_point(aes(color = runner)) +
  labs(title = "Runner-Specific Models with Posterior Samples",
       x = "Age", y = "Net Time")
```



```
# Comparing runner-specific models with the global model
ggplot(running, aes(y = net, x = age, group = runner)) +
  geom_abline(data = runner_summaries_1,
             aes(intercept = runner_intercept, slope = B1), alpha = 1.0, color = "gray") +
  geom_abline(intercept = B0, slope = B1, color = "blue", linewidth=2) +
  lims(x = c(50, 61), y = c(50, 135)) +
  labs(title = "Runner-Specific Models vs Global Model",
       x = "Age", y = "Net Time")
```

Runner-Specific Models vs Global Model



Summarizing variance components

Shows how much variability is attributed to runners vs residuals

```
tidy_sigma <- tidy(running_model_1, effects = "ran_pars")
print(tidy_sigma)
```

```
## # A tibble: 2 x 3
##   term                group  estimate
##   <chr>              <chr>    <dbl>
## 1 sd_(Intercept).runner runner    13.4
## 2 sd_Observation.Residual Residual    5.26
```

```
sigma_0 <- tidy_sigma[1, "estimate"]
sigma_y <- tidy_sigma[2, "estimate"]

# Calculating variance proportions
runner_variance_prop <- sigma_0^2 / (sigma_0^2 + sigma_y^2)
residual_variance_prop <- sigma_y^2 / (sigma_0^2 + sigma_y^2)
list(runner_variance = runner_variance_prop,
     residual_variance = residual_variance_prop)
```

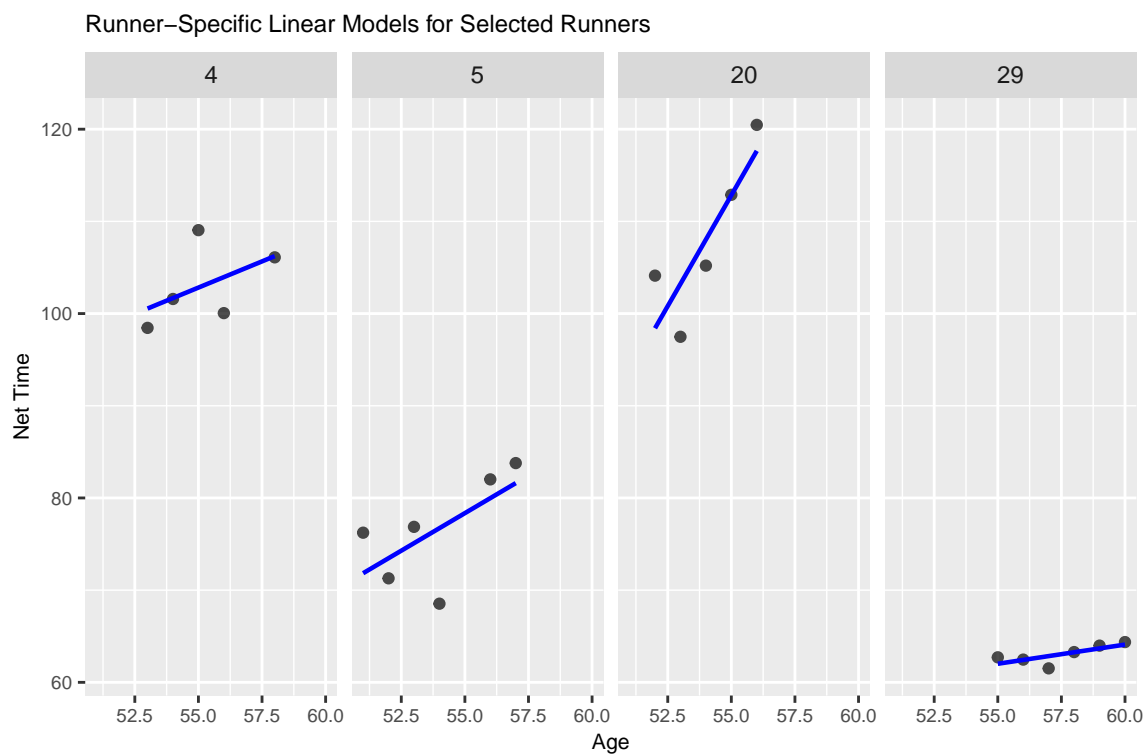
```
## $runner_variance
##   estimate
## 1 0.8661504
##
## $residual_variance
##   estimate
## 1 0.1338496
```

Varying Intercepts and Slopes Model: Analysis and Visualization

Plot runner-specific models in the data for 4 selected runners: Adding a linear trendline for each runner

```
running %>%  
  filter(runner %in% c("4", "5", "20", "29")) %>%  
  ggplot(aes(x = age, y = net)) +  
  geom_point(alpha = 0.7) + # Transparency for better point visibility  
  geom_smooth(method = "lm", se = FALSE, color = "blue", linewidth = 0.8) +  
  facet_grid(~ runner) +  
  labs(title = "Runner-Specific Linear Models for Selected Runners",  
       x = "Age", y = "Net Time")
```

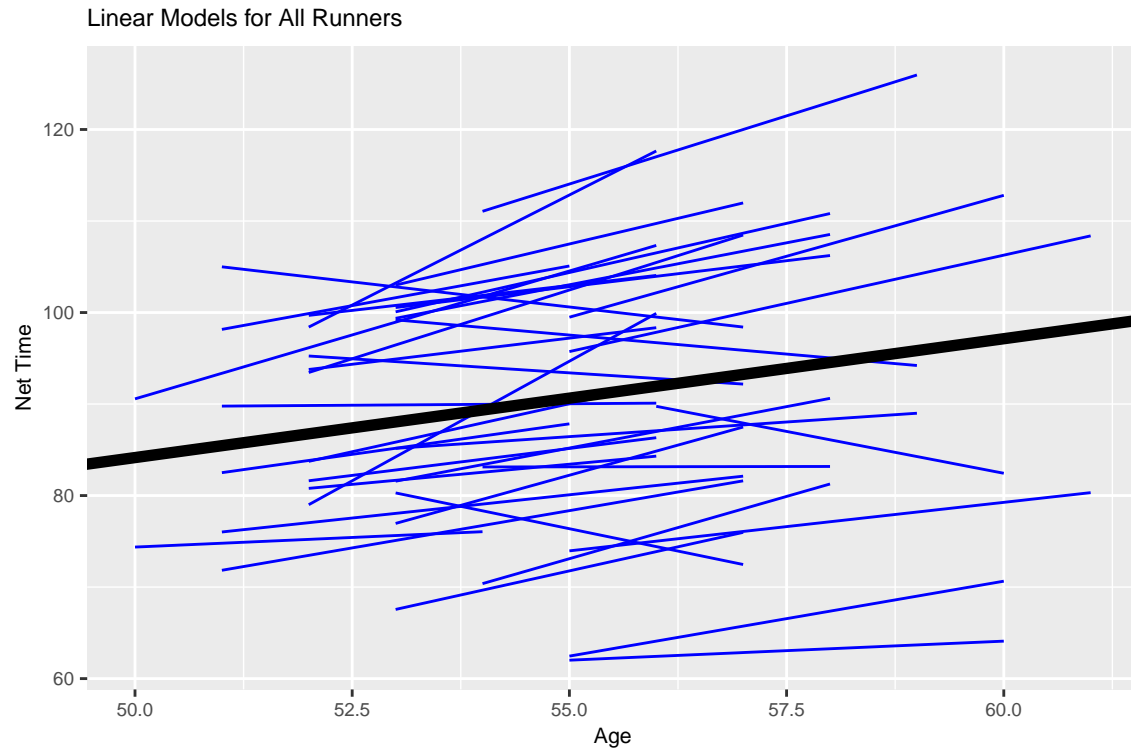
`geom_smooth()` using formula = 'y ~ x'



Plot linear models for all runners: Showing general trends for all runners

```
ggplot(running, aes(x = age, y = net, group = runner)) +  
  geom_smooth(method = "lm", se = FALSE, size = 0.5, color = "blue") +  
  geom_abline(intercept = B0, slope = B1, color = "black", linewidth=2) +  
  labs(title = "Linear Models for All Runners",  
       x = "Age", y = "Net Time")
```

`geom_smooth()` using formula = 'y ~ x'



Fit the varying intercepts and slopes model

Note: Model fitting can be slow, so the number of iterations is reduced compared to the books implementation

```
running_model_2 <- stan_glmer(
  net ~ age + (age | runner),
  data = running, family = gaussian,
  prior_intercept = normal(100, 10),      # Prior for the global intercept
  prior = normal(2.5, 1),                 # Prior for the slope
  prior_aux = exponential(1, autoscale = TRUE), # Prior for residual SD
  prior_covariance = decov(reg = 1, conc = 1, shape = 1, scale = 1), # Prior for covariance
  chains = 1, iter = 2400, adapt_delta = 0.99999 # Reduced chains and iterations for speed
)
```

Summarize global regression parameters: Provides posterior estimates with credible intervals

```
global_summary <- tidy(running_model_2, effects = "fixed", conf.int = TRUE, conf.level = 0.80)
print(global_summary)
```

```
## # A tibble: 2 x 5
##   term      estimate std.error conf.low conf.high
##   <chr>      <dbl>    <dbl>   <dbl>   <dbl>
## 1 (Intercept)  18.8      11.6    3.12   33.3
## 2 age         1.31     0.217   1.04    1.60
```

Extract group-specific parameters: Retrieves MCMC chains for runner-specific intercepts and slopes

```

runner_chains_2 <- running_model_2 %>%
  spread_draws(`(Intercept)`, b[term, runner], `age`) %>%
  pivot_wider(names_from = term, names_glue = "b_{term}",
              values_from = b) %>%
  mutate(runner_intercept = `(Intercept)` + `b_(Intercept)`, # Runner-specific intercept
         runner_age = age + b_age) # Runner-specific slope

# Compute posterior medians for runner-specific models
# Medians summarize central tendencies of intercepts and slopes
runner_summaries_2 <- runner_chains_2 %>%
  group_by(runner) %>%
  summarize(runner_intercept = median(runner_intercept),
            runner_age = median(runner_age))

# Display the first three rows of the summary
head(runner_summaries_2, 3)

## # A tibble: 3 x 3
##   runner runner_intercept runner_age
##   <chr>         <dbl>         <dbl>
## 1 runner:1         18.7           1.06
## 2 runner:10        19.0           1.74
## 3 runner:11        18.8           1.31

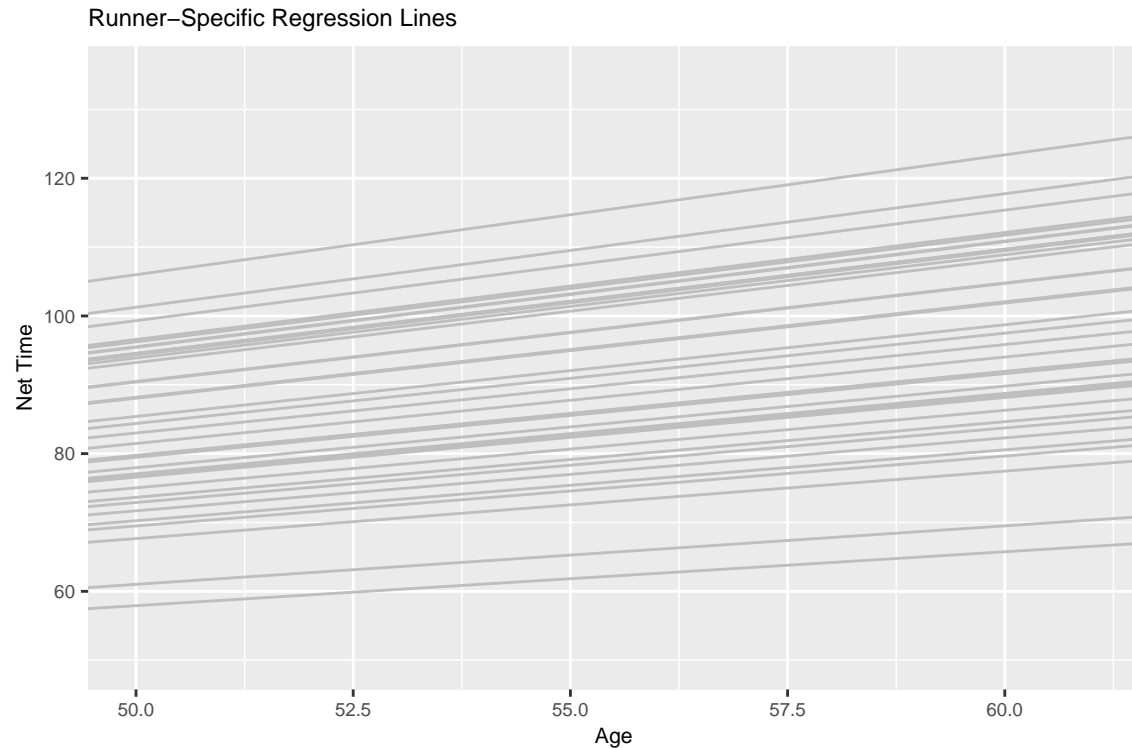
```

Plot regression lines for each runner based on posterior medians: Gray lines represent individual models

```

ggplot(running, aes(y = net, x = age, group = runner)) +
  geom_abline(data = runner_summaries_2, aes(intercept = runner_intercept, slope = runner_age), color =
  lims(x = c(50, 61), y = c(50, 135)) +
  labs(title = "Runner-Specific Regression Lines",
       x = "Age", y = "Net Time")

```

```
# Highlight two runners (demonstrating shrinkage effect)
runner_summaries_2 %>%
  filter(runner %in% c("runner:1", "runner:10"))
```

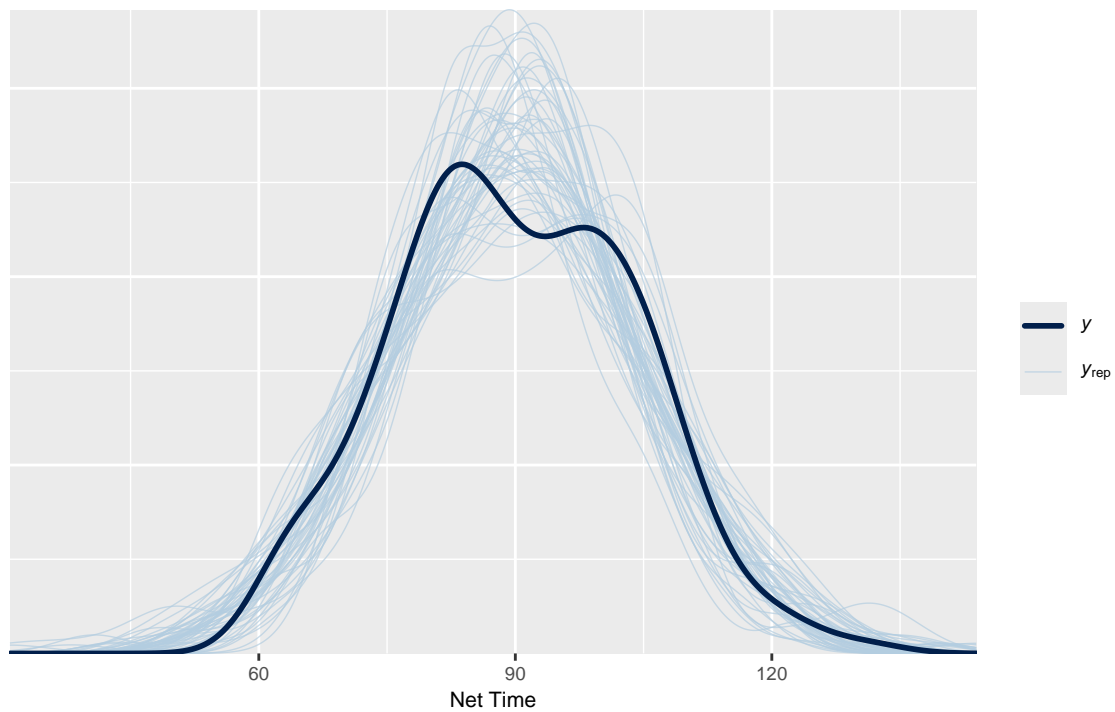
```
## # A tibble: 2 x 3
##   runner runner_intercept runner_age
##   <chr>         <dbl>         <dbl>
## 1 runner:1         18.7         1.06
## 2 runner:10        19.0         1.74
```

Model Selection and Posterior Predictions

```
# Fit a complete pooled model (no runner-level variation)
complete_pooled_model <- stan_glm(
  net ~ age,
  data = running, family = gaussian,
  prior_intercept = normal(0, 2.5, autoscale = TRUE), # Weakly informative prior for intercept
  prior = normal(0, 2.5, autoscale = TRUE),           # Weakly informative prior for slope
  prior_aux = exponential(1, autoscale = TRUE),       # Prior for residual standard deviation
  chains = 4, iter = 2000                           # Reduced iterations for faster computation
)
```

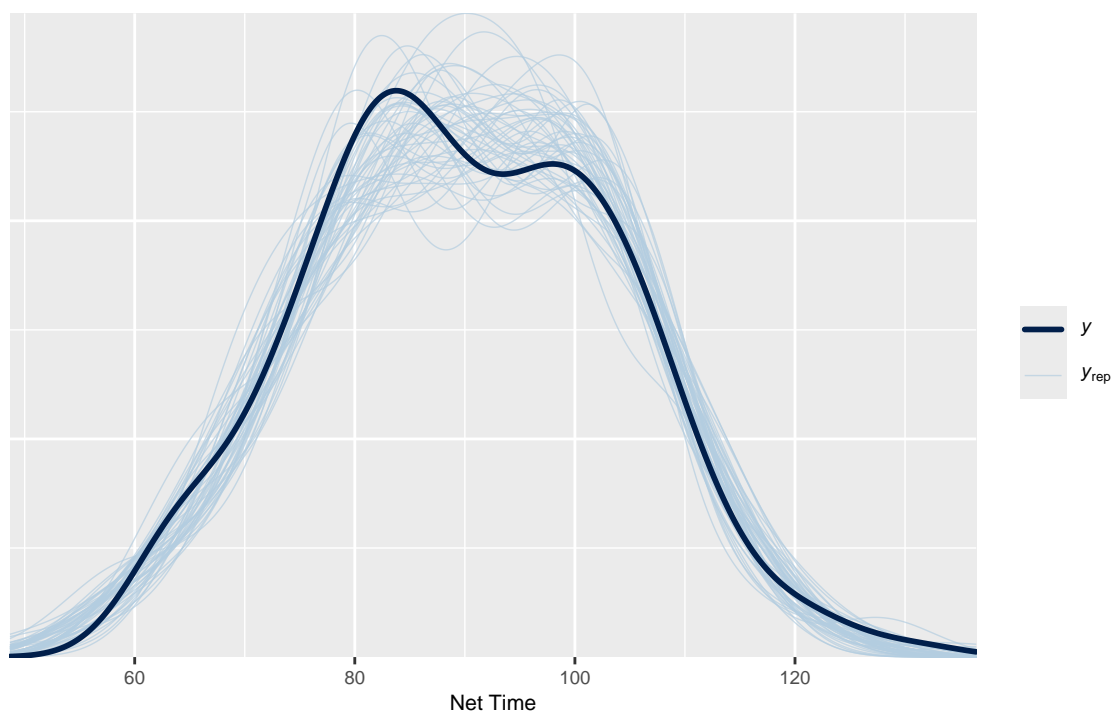
```
# Posterior predictive checks to compare model fit
# Complete pooled model
pp_check(complete_pooled_model) +
  labs(x = "Net Time", title = "Posterior Predictive Check: Complete Pooled Model")
```

Posterior Predictive Check: Complete Pooled Model



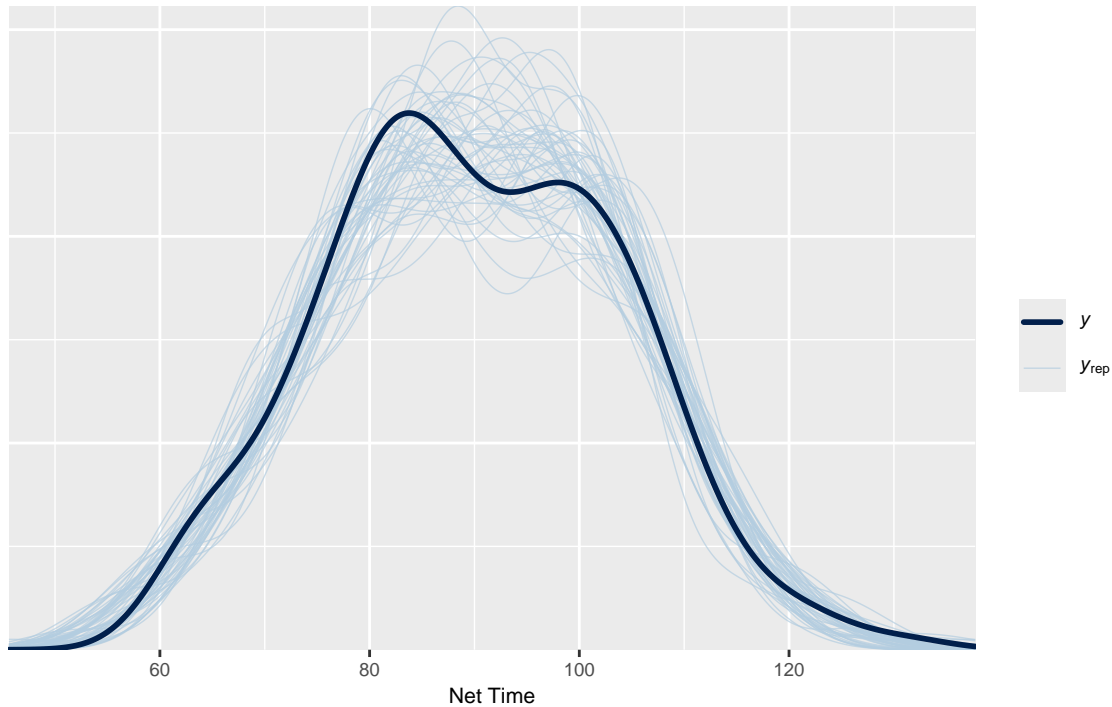
```
# Running model 1 (varying intercepts)
pp_check(running_model_1) +
  labs(x = "Net Time", title = "Posterior Predictive Check: Running Model 1 (Varying Intercepts)")
```

Posterior Predictive Check: Running Model 1 (Varying Intercepts)



```
# Running model 2 (varying intercepts and slopes)
pp_check(running_model_2) +
  labs(x = "Net Time", title = "Posterior Predictive Check: Running Model 2 (Varying Intercepts & Slopes)")
```

Posterior Predictive Check: Running Model 2 (Varying Intercepts & Slopes)



```
# Assessing Predictive Accuracy
# Custom function `prediction_summary` evaluates predictive performance
prediction_summary(model = running_model_1, data = running)
```

```
##           mae mae_scaled within_50 within_95
## 1 2.575586  0.4542772 0.6972973 0.9783784
```

```
prediction_summary(model = running_model_2, data = running)
```

```
##           mae mae_scaled within_50 within_95
## 1 2.642669  0.4605346 0.7027027 0.972973
```

```
# Cross-validation metrics (using LOO)
# Note: Leave-One-Out cross-validation can be computationally intensive
elpd_hierarchical_1 <- loo(running_model_1)
elpd_hierarchical_2 <- loo(running_model_2)
```

```
# Print ELPD estimates for model comparison
print(elpd_hierarchical_1$estimates)
```

```
##           Estimate      SE
## elpd_loo -589.04647 16.797491
## p_loo      35.40028  6.243469
## looic      1178.09295 33.594982
```

```
print(elpd_hierarchical_2$estimates)
```

```
##           Estimate      SE
## elpd_loo -588.93419 17.258979
## p_loo    37.30624  6.586819
## looic    1177.86838 34.517958
```

Posterior Prediction for Specific Ages and Runners

Predict race times at age 61 for two runners and a new individual Using the varying-intercepts model (Running Model 1)

```
predict_next_race <- posterior_predict(
  running_model_1,
  newdata = data.frame(
    runner = c("1", "Miles", "10"), # Existing runners 1 & 10, plus a new individual "Miles"
    age = c(61, 61, 61)             # Predicting for age 61
  )
)

# Display posterior means for predicted times
predicted_means <- colMeans(predict_next_race)
names(predicted_means) <- c("Runner 1", "Miles (new)", "Runner 10")
print(predicted_means)
```

```
##      Runner 1 Miles (new)  Runner 10
##      84.64629   98.74008   123.04187
```

```
# Posterior predictive distributions
mcmc_areas(predict_next_race, prob = 0.8) +
  ggplot2::scale_y_discrete(labels = c("Runner 1", "Miles (new)", "Runner 10")) +
  labs(title = "Posterior Predictive Distributions at Age 61",
       x = "Predicted Net Time", y = "Individuals")
```

```
## Scale for y is already present.
```

```
## Adding another scale for y, which will replace the existing scale.
```

