Bayesian model checking

Stat 340: Bayesian Statistics

Coffee ratings

- Interest is in modeling coffee ratings on a 0–100 scale
- Possible predictors include grades on features such as its aroma, aftertaste, flavor, etc.
- Data available in `bayesrules` as `coffee_ratings`

```
library(bayesrules)
data("coffee_ratings")
coffee_ratings <- coffee_ratings %>%
  select(farm_name, total_cup_points, aroma, aftertaste)
```

Data originally from [TidyTuesday](), contributed by James LeDoux

Your turn 1

Suppose we want to fit a model with the following sampling model:

$$\begin{aligned} Y_i | \beta_0, \beta_1, \sigma &\overset{ind}{\sim} \mathcal{N}(\mu_i, \sigma) \\ \mu_i &= \beta_0 + \beta_1 \texttt{aroma} \end{aligned}$$

What assumptions/conditions does this model require?

Your turn 2

The coffee_ratings data includes ratings and features of 1339 different batches of beans grown on 571 different farms.

Explain why using this data to model ratings by aroma likely violates the independence assumption.

Updated example

Let's suppose we only have one observation per farm

```
set.seed(84735)
new_coffee <- coffee_ratings %>%
  group_by(farm_name) %>%
  sample_n(1) %>%
  ungroup()
```

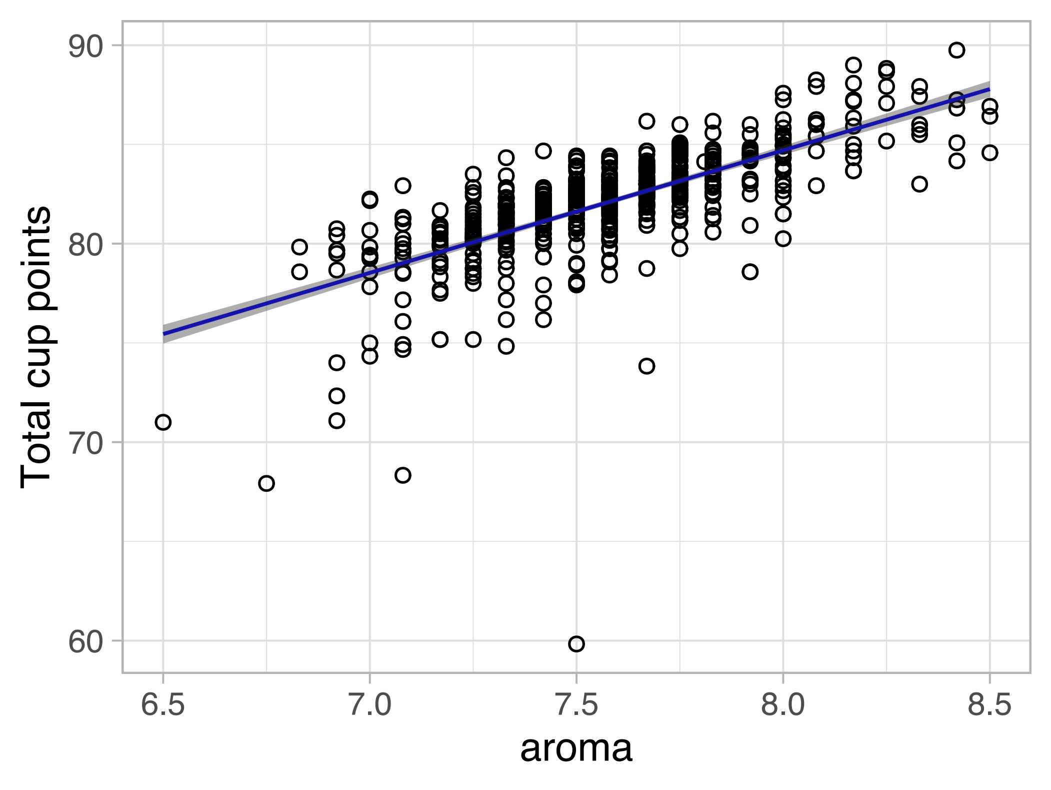
```
## # A tibble: 572 × 4
##   farm_name                            total_cup_points aroma aftertaste
##   <fct>                                           <dbl> <dbl>      <dbl>
## 1 -                                                84    7.67       7.67
## 2 1                                                76.2  7.33       6.5 
## 3 200 farms                                        67.9  6.75       6.42
## 4 2000 farmers                                     72.3  6.92       7.08
## 5 2000 farms                                       80.8  7.42       7.42
## 6 a shu she coffee 阿束社咖啡莊園                  80.1  7.25       7.25
## # … with 566 more rows
```

Weakly informative prior model

```
##
## JAGS model summary statistics from 15000 samples (thin = 80; chains = 3; ad
##
##        Lower95 Median Upper95 Mean     SD Mode     MCerr MC%ofSD SSeff   AC.80
## beta0     31.4   35.4    39.1 35.3   1.95 35.4    0.0603     3.1  1044    0.26
## beta1     5.67   6.17    6.68 6.17  0.257 6.16   0.00788     3.1  1061    0.26
## sigma     1.84   1.95    2.07 1.96 0.0577 1.95  0.000473     0.8 14887  0.0013
##
##        psrf
## beta0 1.01
## beta1 1.01
## sigma    1
##
## Total time taken: 23.1 seconds
```

Your turn 3

Based on this plot of the fitted model, do you see any issues with our assumptions?



Residual plots

- In previous courses, you used residual plots to diagnose linearity, constant variance assumptions
- $e_i = y - (\widehat{\beta}_0 + \widehat{\beta}_1 x_i)$
- To create a Bayesian analog, use posterior point estimates for $\widehat{\beta}_0$ and $\widehat{\beta}_1$

Residual plots

If you run multiple chains, `tidybayes::tidy_draws` makes it easy to combine them into a single data frame

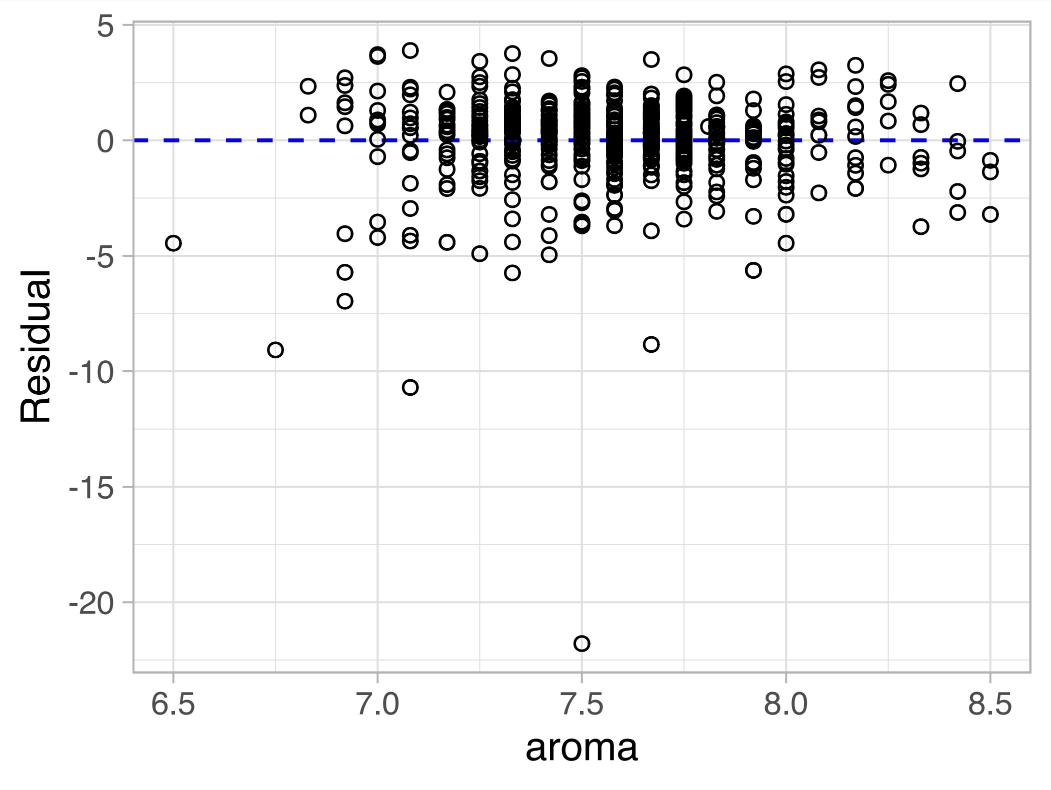
```
post_draws <- tidybayes::tidy_draws(slr_draws$mcmc)
```

To calculate $\widehat{\beta}_0 + \widehat{\beta}_1 x_i$ and e_i, I find it easiest to add columns to the data

```
resids <- new_coffee %>%
  mutate(
    beta0 = median(post_draws$beta0),
    beta1 = median(post_draws$beta1),
    yhat  = beta0 + beta1 * aroma,
    resid = total_cup_points - yhat
  )
```

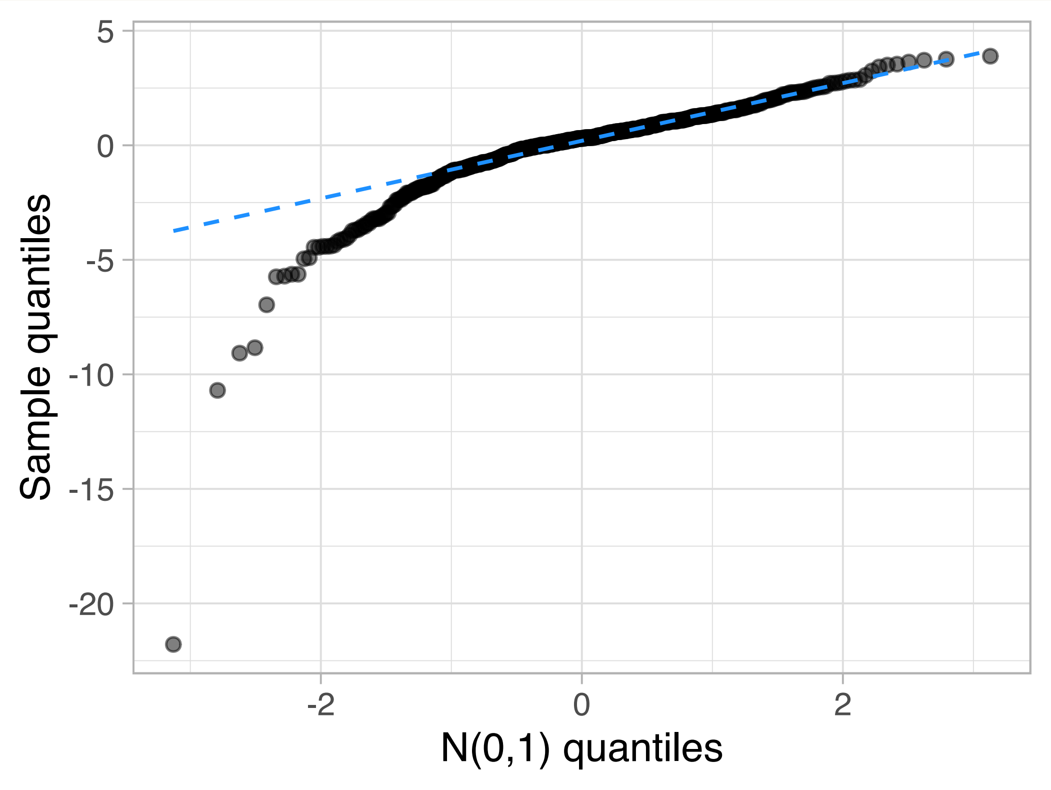
Residual plots

Based on this residual plot, are there any model deficiencies?



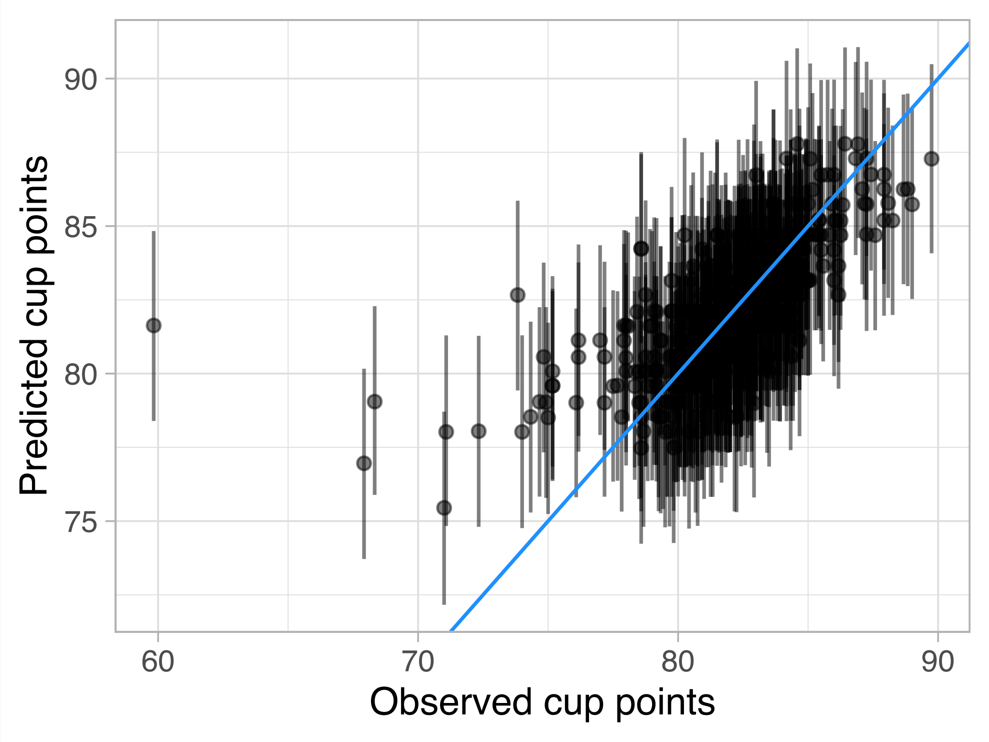
Residual normal Q-Q plot

We can make other familiar plots of residuals using these posterior estimates



Posterior prediction plots

Plot predicted y (with intervals) vs. observed y and explore discrepancies



For each simulated $(\beta_0^{(j)}, \beta_1^{(j)}, \sigma^{(j)})$

- Calculate $\mu_i^{(j)} = \beta_0^{(j)} + \beta_1^{(j)} x_i$
- Simulate $\tilde{y}_i^{(j)}$ from $\mathcal{N}(\mu_i^{(j)}, \sigma^{(i)})$

Posterior prediction plots

Create a function to calculate μ_i given an x_i value

```
mu_link <- function(x) {
  post_draws[["beta0"]] + post_draws[["beta1"]] * x
}
```

Calculate S μ_i's for each values in the x vector (`new_coffee$aroma` here)

```
mu_draws<- sapply(new_coffee$aroma, mu_link)
```

Now, we have a $S \times n$ matrix, each column corresponds to an x_i

Posterior prediction plots

Simulate one $\tilde{y}_i$ for each μ_i in each column

```
S <- nrow(post_draws)
y_draws <- apply(mu_draws, 2, function(x) rnorm(S, x, post_draws[["sigma"]]))
```

Now, we have a $S \times n$ matrix of simulated observations

Calculate the mean and PI for each column

```
y_means <- colMeans(y_draws)
y_pis   <- apply(y_draws, 2, quantile, probs = c(0.05, 0.95))
```

Posterior prediction plots

To use `ggplot2`, let's combine the results into a data frame

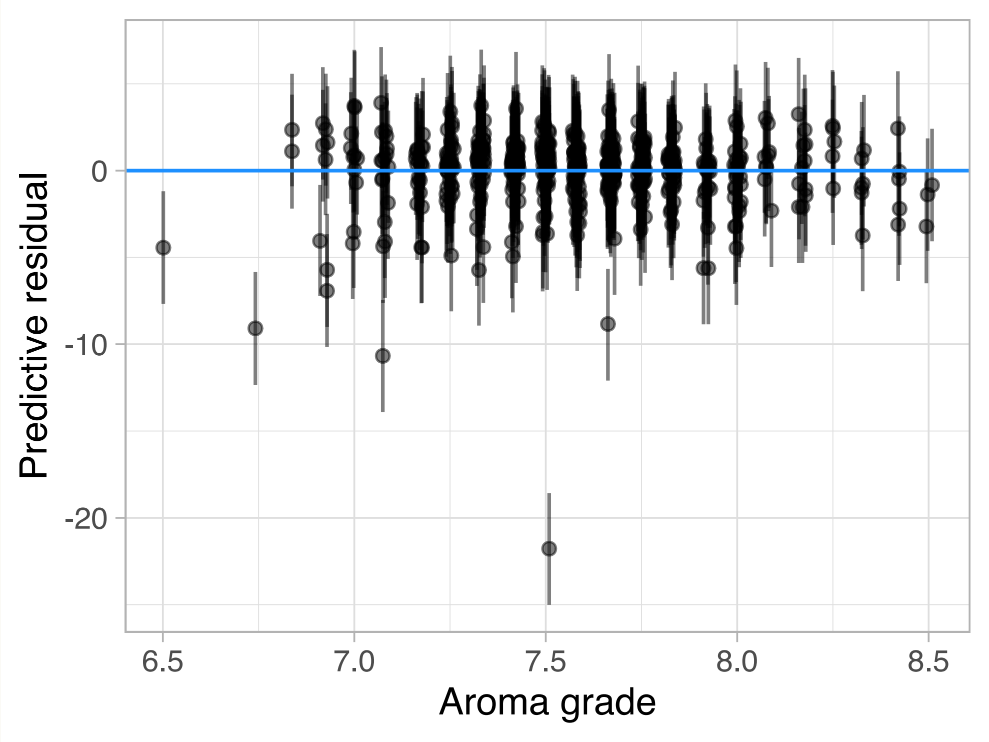
```
post_pred_data <- data.frame(
  y = new_coffee$total_cup_points, # original y
  y_pred = y_means,                # avg. predicted response
  y_lo = y_pis[1,],                # lower bound of predicted response
  y_hi = y_pis[2,]                 # upper bound of predicted response
)
```

Render the plot

```
ggplot(post_pred_data, aes(x = y)) +
  geom_point(aes(y = y_means), alpha = 0.5) +
  geom_linerange(aes(ymin = y_lo, ymax = y_hi), alpha = 0.5) +
  geom_abline(slope = 1, intercept = 0, color = "dodgerblue")
```

Predictive residuals

Plot residual interval estimates vs. predictor



- Does this plot highlight any deficiencies of our model?
- Is it easier to read than the posterior prediction plot?

Predictive residuals

We can start with our posterior prediction plot data set, then calculate the midpoint and endpoints of our residual intervals using those prediction intervals

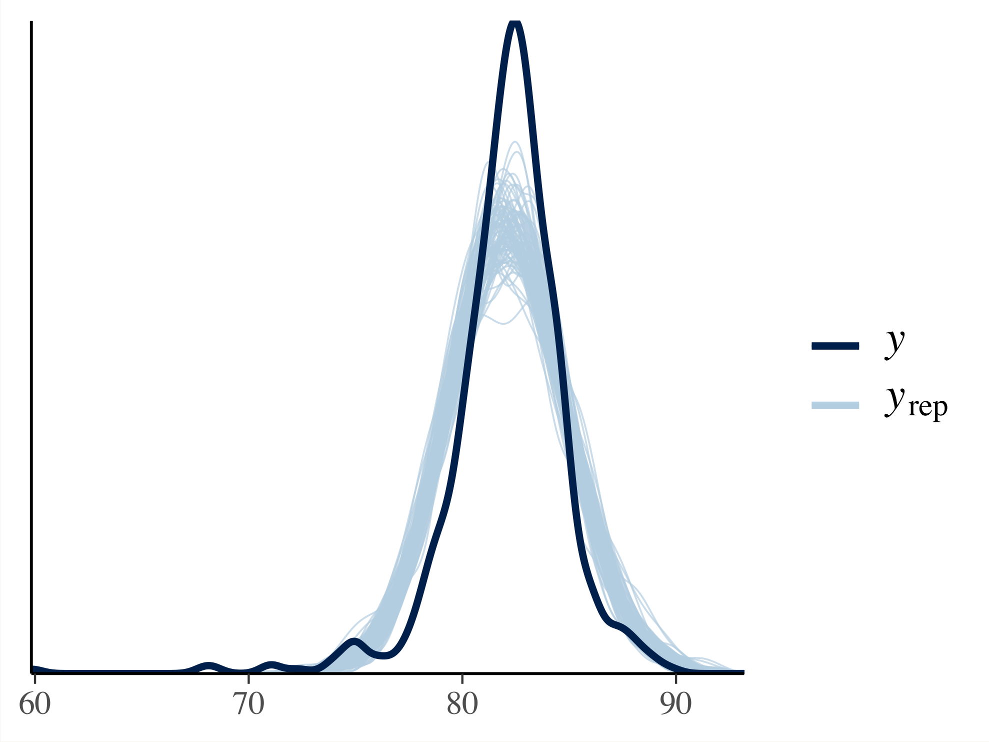
```
pred_resids <- post_pred_data %>%
  mutate(
    resid = y - y_pred,
    resid_lo = y - y_lo,
    resid_hi = y - y_hi,
    x = jitter(new_coffee$aroma, factor = 2.5) # avoiding some overlap
)
```

Create the plot

```
ggplot(pred_resids, aes(x = x)) +
  geom_point(aes(y = resid), alpha = 0.5) +
  geom_linerange(aes(ymin = resid_lo, ymax = resid_hi), alpha = 0.5) +
  geom_hline(yintercept = 0, color = "dodgerblue")
```

Posterior predictive checks

We can also examine the density of the observed and predicted responses.



- Generate a new data set for each $(\beta_0^{(j)}, \beta_1^{(j)}, \sigma^{(j)})$ for all x_i in the data set
- Same process as with posterior prediction plots, different display
- A good model should for a data set should be able to generate data similar to the observed data

Informative priors

Centering approach

Previously, we've seen that **centering the predictor** around either a sample or hypothesized value **makes the intercept interpretable**

Example:

- On an average temperature day, say 65 or 70 degrees for D.C., there are typically around 5000 riders, though this average could be somewhere between 3000 and 7000.
- For every one degree increase in temperature, ridership typically increases by 100 rides, though this average increase could be as low as 20 or as high as 180.
- At any given temperature, daily ridership will tend to vary with a moderate standard deviation of 1250 rides.

Standardization approach

Standardizing both the response and predictor also helps with interpretation

$$y_i^* = \frac{y_i - \overline{y}}{s_y}, \qquad x_i = \frac{x_i - \overline{x}}{s_x}$$

Updated model: $Y_i^*|\mu_i^*, \sigma \overset{ind}{\sim} \mathcal{N}(\mu_i, \sigma), \qquad \mu_i^* = \beta_0 + \beta_1 x_i^*$

Interpretations:

Standardization example

How can we specify our priors for β_0 and β_1 based on the following prior information?

- On an average temperature day, say 65 or 70 degrees for D.C., there are typically around 5000 riders, though this average could be somewhere between 3000 and 7000.
- The correlation between ridership and temperature is expected to be of moderate strength, most likely between 0.45 and 0.75.

Conditional means approach

- So far we have only tried to elicit prior information directly on the parameters
- When working with some domain experts, it might be easier to elicit this information indirectly
- We can ask about the means, μ_i^* and μ_j^*, for predictor values x_i and x_j
- Now we have two points (x_i, μ_i^*) and (x_j, μ_j^*)

Conditional means example

How can we specify our priors for β_0 and β_1 based on the following prior information?

- On a 70 $^\circ$F day there are typically around 4000 riders, though this average could be somewhere between 3000 and 5000.
- On a 50 $^\circ$F day there are typically around 2000 riders, though this average could be somewhere between 1200 and 2800.

Inducing priors in JAGS

To use an induced prior, tell JAGS how to calculate the *original* parameter from the alternative parameterization (e.g., from μ_1 and μ_2)

```
modelString = "model{

# Sampling model
for(i in 1:n) {
  y[i] ~ dnorm(beta0 + beta1 * x[i], invsigma2)
}

# Prior models
mu1 ~ dnorm(m1, s1)                              # Specify prior info
mu2 ~ dnorm(m2, s2)                              # Specify prior info
beta1 <- (mu2 - mu1) / (x2 - x1)                 # Calc. beta1 from mu1 & mu2
beta0 <- mu1 - x1 * (mu2 - mu1) / (x2 - x1) # Calc. beta0 from mu1 & mu2

invsigma2 ~ dgamma(1, 1)
sigma <- pow(invsigma2, -1/2)
}"
```