hierarchical_milk_bayes

2026-01-16

```r
# Load packages
library(bayesrules)
library(rstanarm)
```

```
## Loading required package: Rcpp

## This is rstanarm version 2.32.1

## - See https://mc-stan.org/rstanarm/articles/priors for changes to default priors!

## - Default priors may change, so it's safest to specify priors, even if equivalent to the defaults.

## - For execution on a local, multicore CPU with excess RAM we recommend calling

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

```r
library(bayesplot)
```

```
## This is bayesplot version 1.12.0

## - Online documentation and vignettes at mc-stan.org/bayesplot

## - bayesplot theme set to bayesplot::theme_default()

##    * Does _not_ affect other ggplot2 plots

##    * See ?bayesplot_theme_set for details on theme setting
```

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.2     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(tidybayes)
library(broom.mixed)
library(ggplot2)
library(purrr)
```

```r
weekly <- read.csv("weekly_tea.csv")
weekly$holiday = as.factor(weekly$holiday)
weekly <- na.omit(weekly)
```

Hierarchical models are useful for modeling grouped data.

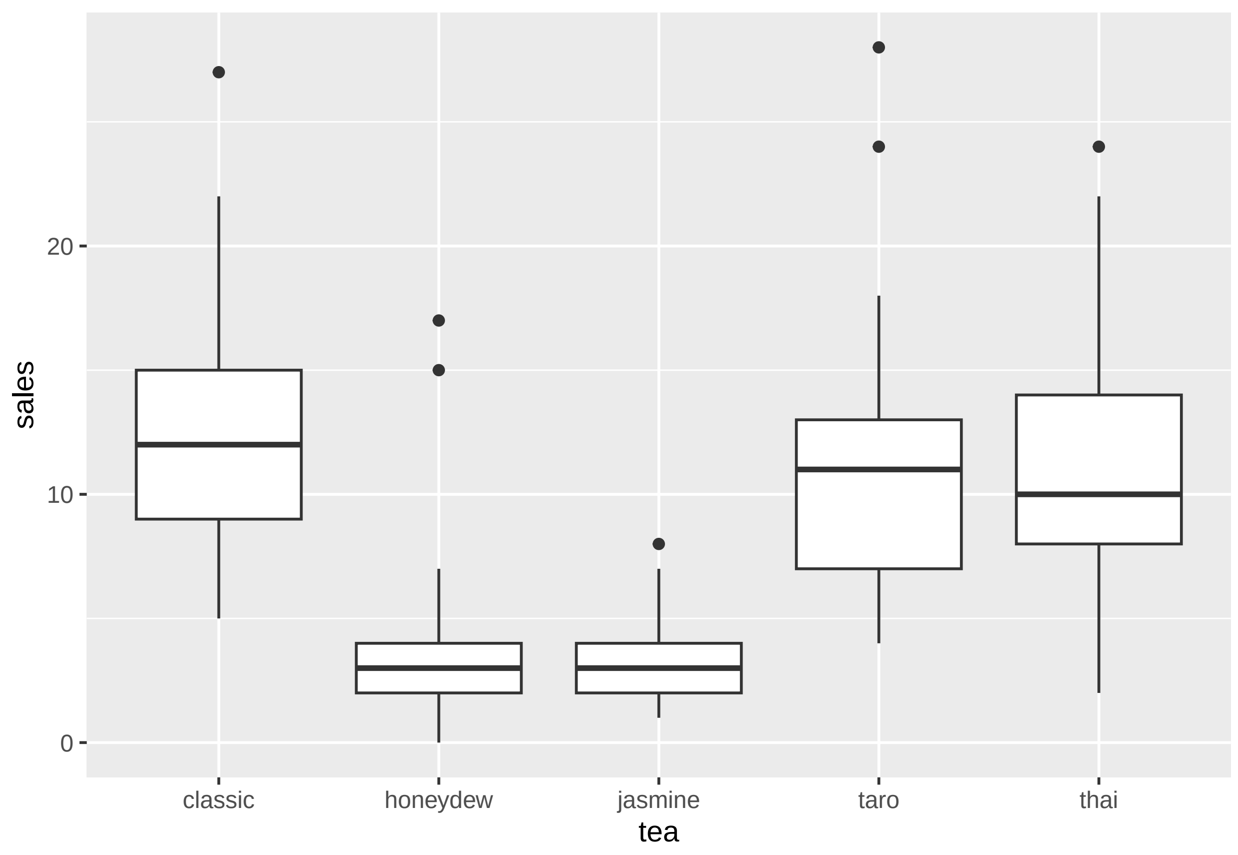
In our case, we are trying to predict the weekly sales for different types of milk tea. This can be useful for planning inventory, since different types of milk tea require different ingredients.

Why might a hierarchical model be suitable in this case?

In our previous model, we used tea as a predictor, resulting in different intercepts per tea. Note that the coefficients for sales_lag1 and weekend_holiday were shared, however. This approach worked for predicting tea sales broadly, since there were only two possible categories of interest: green tea and milk tea. sales ~ sales_lag1 + weekend_holiday + tea

For our new problem, however, a hierarchical model seems more suitable. Since we are looking at the same type of product (milk teas), it could very well be the case that tea effects are more closely related and they come from a common distribution. Incorporating a hierarchical structure in our model can help us capture any existing dependencies between teas and produce more realistic predictions. Additionally, our sample of milk teas are not the only types of milk teas out there. If we were to introduce a new type of milk tea, our hierarchical model would be able to draw information from our existing types of milk teas to make sales predictions, which would not be possible with a fixed effects model.

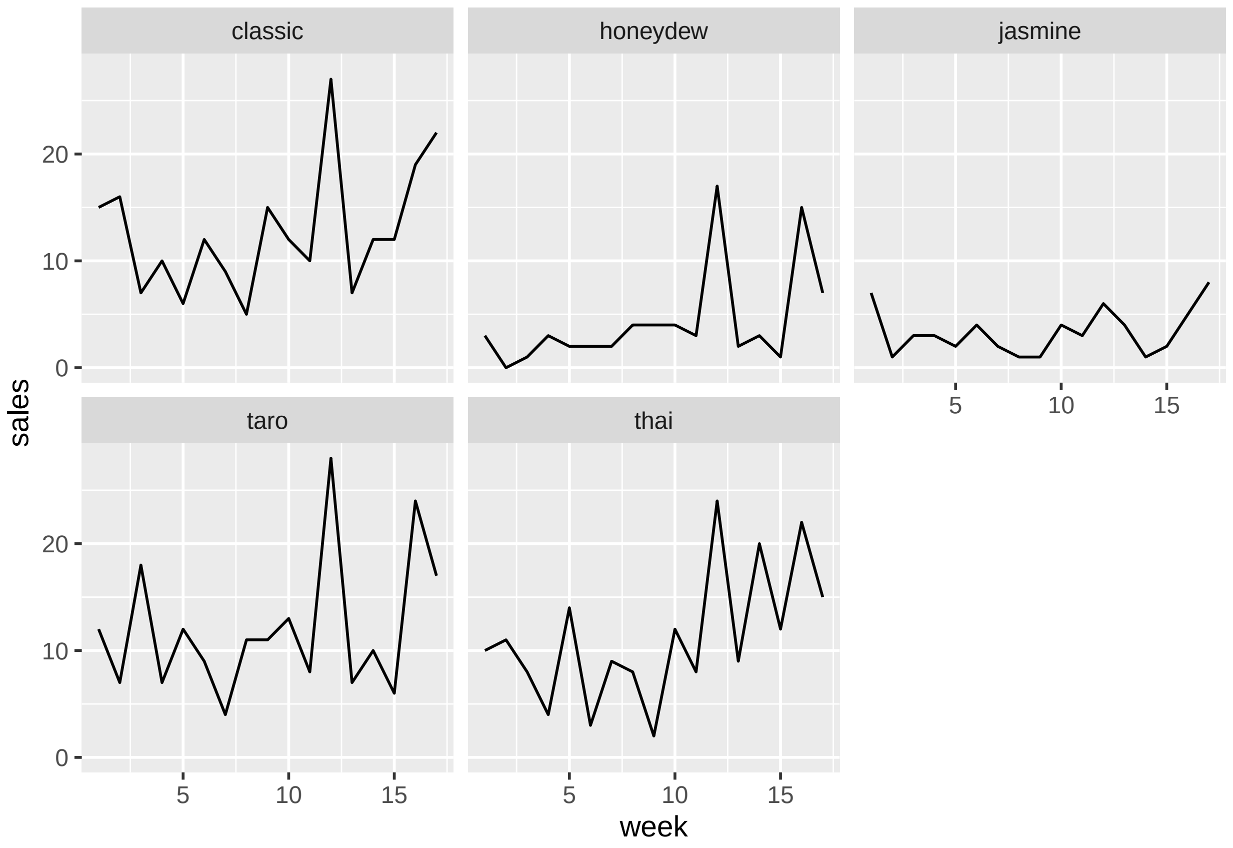
```r
ggplot(weekly, aes(x = tea, y = sales)) +
  geom_boxplot()
```



From the plot, we can see the variability within and between different types of milk teas. For example, we can see that on average, honeydew and jasmine sales are lower than the other types. We can also see that taro sales and Thai sales have very similar variability.

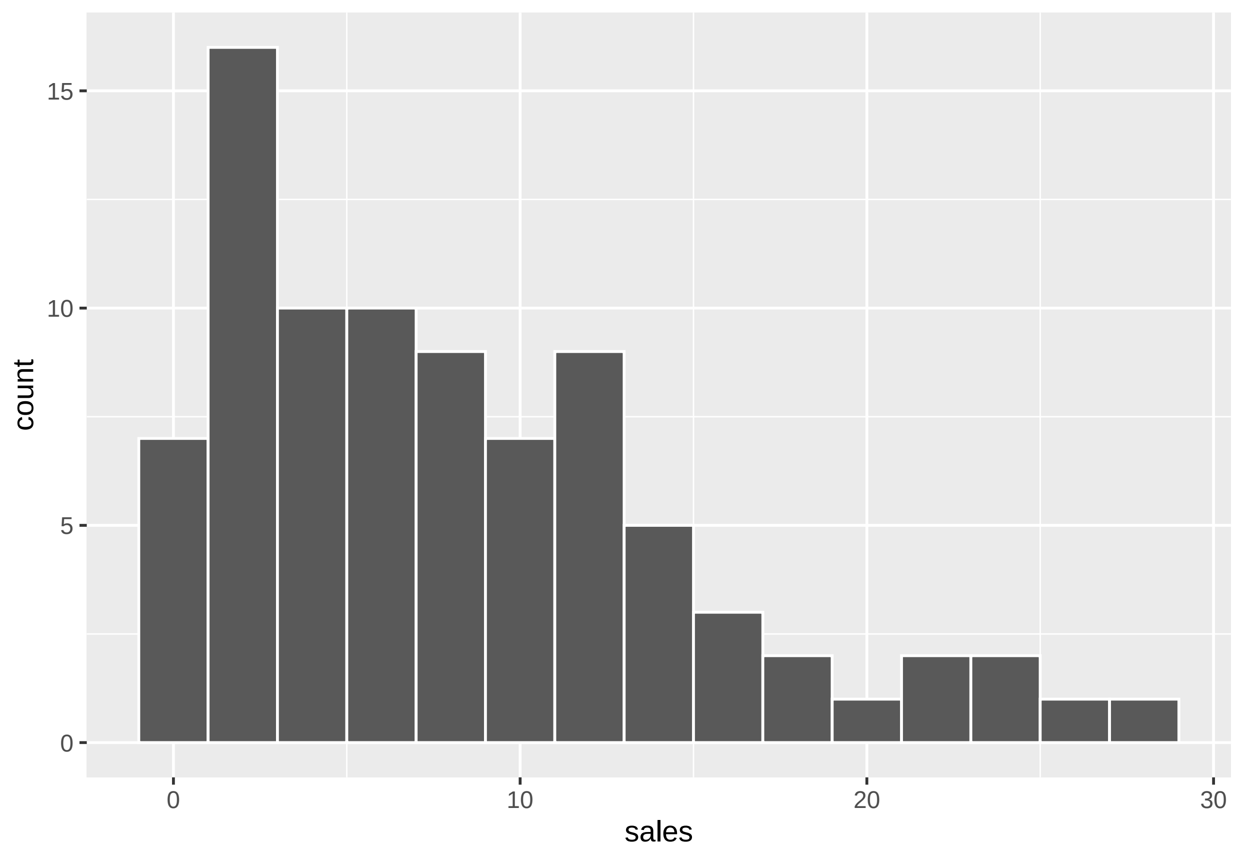
We can also look at how tea sales have changed over time. The teas seem to show similar patterns, with spikes around weeks 12 and 16.

```r
ggplot(weekly, aes(x = week, y = sales)) +
  geom_line() +
  facet_wrap(~ tea)
```



As before, to predict this week's sales, we can look at variables like last week's number of customers, last week's sales, and whether this week contains a holiday.

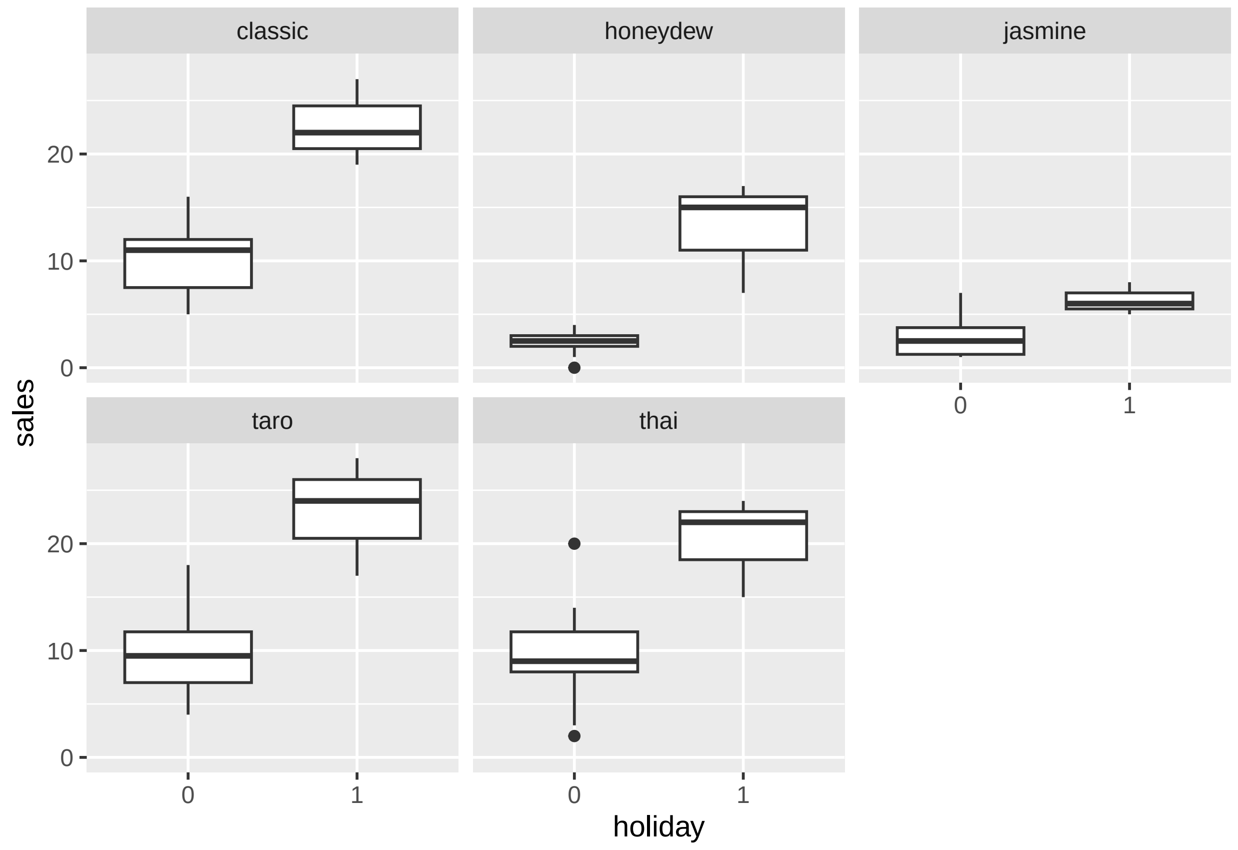
```
ggplot(weekly, aes(x = sales)) +
  geom_histogram(color = "white", bins=15)
```



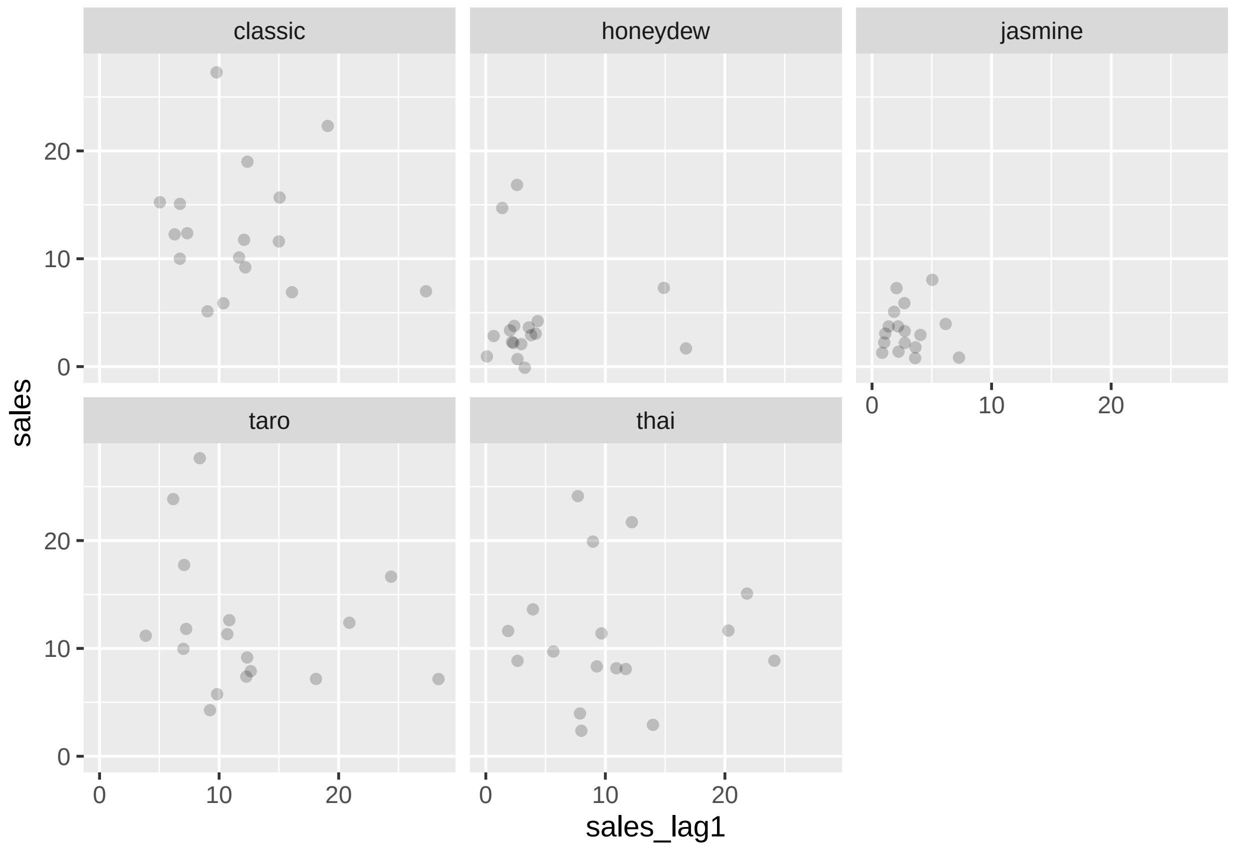
```r
  facet_wrap(~ tea)
```

```
## <ggproto object: Class FacetWrap, Facet, gg>
##     compute_layout: function
##     draw_back: function
##     draw_front: function
##     draw_labels: function
##     draw_panels: function
##     finish_data: function
##     init_scales: function
##     map_data: function
##     params: list
##     setup_data: function
##     setup_params: function
##     shrink: TRUE
##     train_scales: function
##     vars: function
##     super:  <ggproto object: Class FacetWrap, Facet, gg>
```

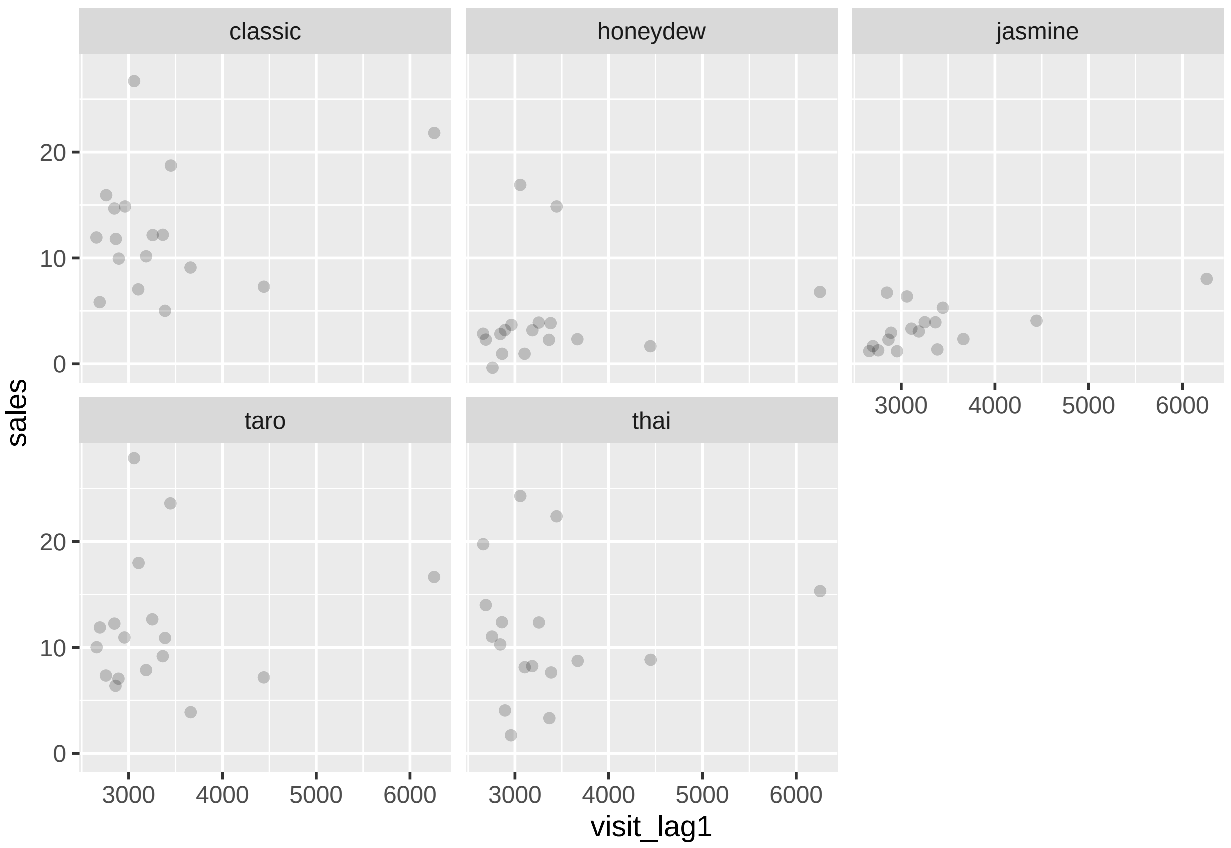
```r
ggplot(weekly, aes(x = holiday, y = sales)) +
  geom_boxplot() +
  facet_wrap(~ tea)
```



```
ggplot(weekly, aes(x = sales_lag1, y = sales)) +
  geom_jitter(alpha=0.2) +
  facet_wrap(~ tea)
```



```r
ggplot(weekly, aes(x = visit_lag1, y = sales)) +
  geom_jitter(alpha=0.2) +
  facet_wrap(~ tea)
```



The data looks right-skewed with most weeks having few sales of an individual type of milk tea and a few weeks having very many sales of an individual type of milk tea. Across all teas, sales are higher on weeks containing holidays. We can see that the sales for certain types of tea are consistently higher or lower than others. For example, classic tea has higher sales than honeydew tea across all levels of visit_lag1.

Looking at the scatterplot comparing this week's sales and last week's sales, there doesn't seem to be a very strong relationship across the tea types. As for this week's sales and last week's customers, some teas exhibit a stronger relationship than others. For example, there is an apparent positive relationship between these two values for jasmine tea. For some other teas like classic or taro, this relationship is harder to make out.

Since our data are nonnegative counts, we can start by considering a Poisson model for our sales.

Let Y_{ij} denote the sales for week i and tea type j, and let $X = (X_{ij1}, X_{i2}, X_{i3})$ denote last week's type-specific tea sales, last week's number of customers, and whether this week contains a holiday.

Then the hierarchical Poisson model can be written as

$$Y_{ij}|\beta_{0j}, \beta_1, \beta_2, \beta_3 \sim Pois(\lambda_{ij})$$

where

$$\log(\lambda_{ij}) = \beta_{0j} + \beta_1 X_{ij1} + \beta_2 X_{i2} + \beta_3 X_{i3}$$

In a hierarchical model, the mean tea sales for a given tea j depends on a tea-specific intercept parameter β_{0j}. However, unlike in non-hierarchical modeling where each intercept is given its own separate prior, each of these intercept parameters are related to each other by shared global parameters β_0 and σ_0.

We have

$$\beta_{0j}|\beta_0, \sigma_0 \overset{ind}{\sim} N(\beta_0, \sigma_0^2)$$

β_0 can be thought of as the average weekly sales considering all tea types. σ_0 is a measure of how spread apart average tea sales of different tea types are from each other.

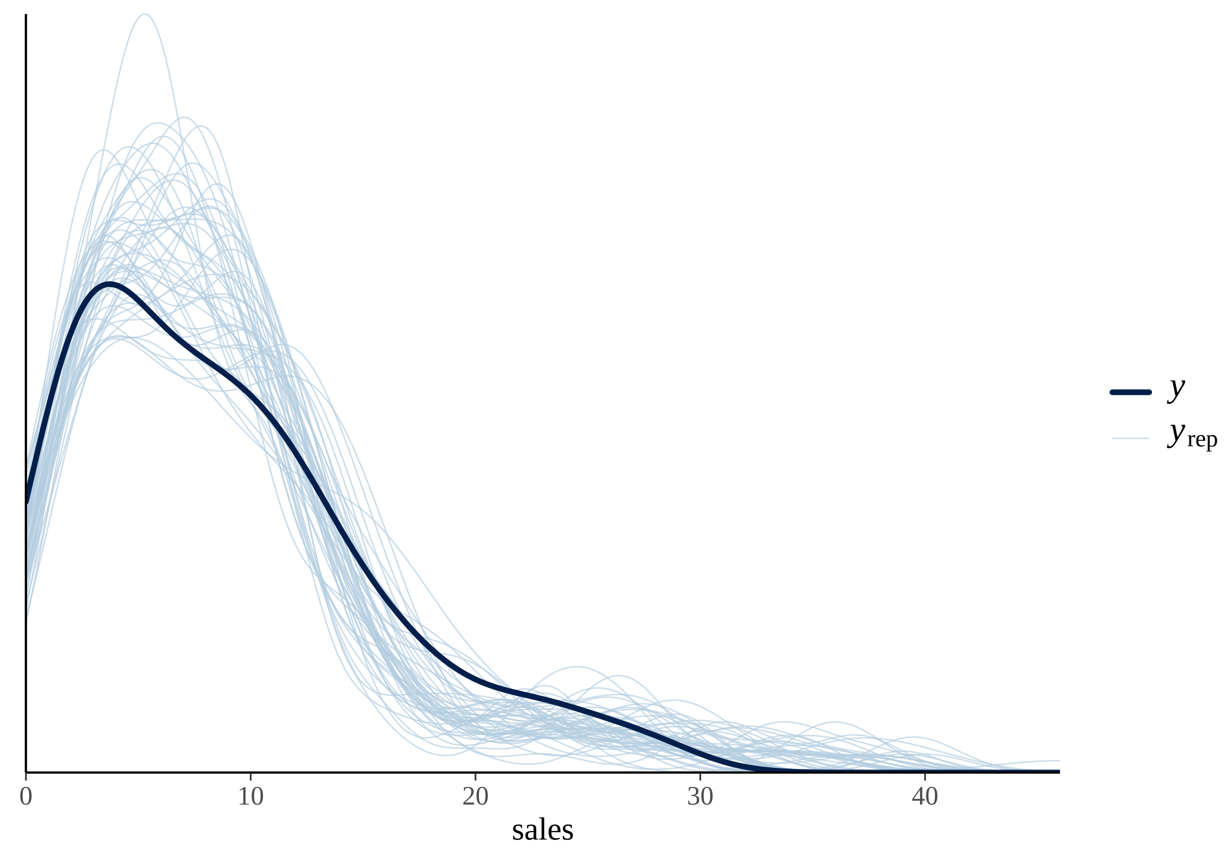
For each global parameter, we need to specify a prior. These global parameters include the aforementioned β_0 and σ_0, but also the coefficients β_1, β_2, and β_3.

There is a restriction that $\sigma_0 > 0$, so we use an Exponential prior for this parameter. For the other parameters, we can use Normal priors. Using the prior assumption that the average weekly sales for one type of tea is somewhere between 7 and 20, we can specify a prior for β_{0c} (the subscript indicates the centered intercept). For simplicity, we can use default priors for the other parameters.

$$\beta_{0c} \sim N(2.5, 0.25^2)$$
$$\beta_k \sim N(0, s_k^2), k \in 1, 2, 3$$
$$\sigma_0 \sim Exp(1)$$

```r
pois_model1 <- stan_glmer(
  sales ~ sales_lag1 + visit_lag1 + holiday + (1 | tea),
  data = weekly, family = poisson,
  prior_intercept = normal(2, .25, autoscale = TRUE),
  prior = normal(0, 2.5, autoscale = TRUE),
  prior_covariance = decov(reg = 1, conc = 1, shape = 1, scale = 1),
  chains = 4, iter = 5000*2, seed = 84735, refresh = 0
)
```

```r
pp_check(pois_model1) +
  xlab("sales")
```



The simulated data are not too far from the actual data. It looks like our model overestimates low weekly sales around 8 and underestimates high weekly sales around 20. It also suggests very high weekly sales around 40 are possible.

Our post posterior check tells us that this model is not too bad. However, we should try other models to see if we can make an improvement. In addition to Poisson models, we can try negative binomial models, which can deal with overdispersed data.

```r
pois_model2 <- stan_glmer(
  sales ~ sales_lag1 + holiday + (1 | tea),
  data = weekly, family = poisson,
  prior_intercept = normal(2, .25, autoscale = TRUE),
  prior = normal(0, 2.5, autoscale = TRUE),
  prior_covariance = decov(reg = 1, conc = 1, shape = 1, scale = 1),
  chains = 4, iter = 5000*2, seed = 84735, refresh = 0
)
```

```r
negb_model1 <- stan_glmer(
  sales ~ sales_lag1 + visit_lag1 + holiday + (1 | tea),
  data = weekly, family = neg_binomial_2,
  prior_intercept = normal(2, .25),
  prior = normal(0, 2.5, autoscale = TRUE),
  prior_aux = exponential(1, autoscale = TRUE),
  prior_covariance = decov(reg = 1, conc = 1, shape = 1, scale = 1),
  chains = 4, iter = 5000*2, seed = 84735, refresh = 0
)
```

```r
negb_model2 <- stan_glmer(
  sales ~ sales_lag1 + holiday + (1 | tea),
  data = weekly, family = neg_binomial_2,
  prior_intercept = normal(2, .25),
  prior = normal(0.5, 2.5, autoscale = TRUE),
  prior_aux = exponential(1, autoscale = TRUE),
  prior_covariance = decov(reg = 1, conc = 1, shape = 1, scale = 1),
  chains = 4, iter = 5000*2, seed = 84735, refresh = 0
)
```

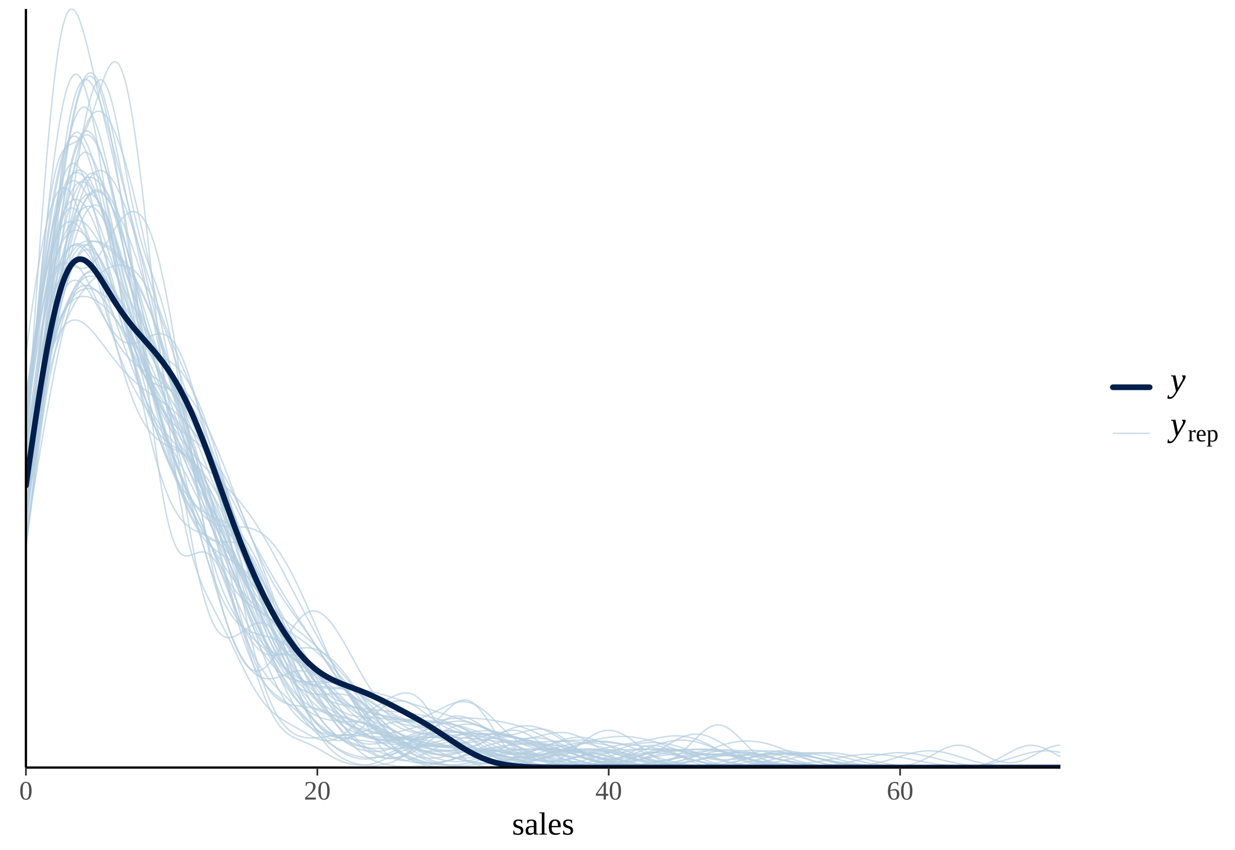
```r
# Calculate ELPD for the 4 models
set.seed(84735)
loo_1 <- loo(pois_model1)
loo_2 <- loo(pois_model2)
loo_3 <- loo(negb_model1)
loo_4 <- loo(negb_model2)

# Results
c(loo_1$estimates[1], loo_2$estimates[1],
  loo_3$estimates[1], loo_4$estimates[1])
```

```
## [1] -215.6709 -215.3833 -220.3394 -219.8607
```

```r
loo_compare(loo_1, loo_2, loo_3, loo_4)
```

```
##             elpd_diff se_diff
## pois_model2  0.0       0.0   
## pois_model1 -0.3       1.3   
## negb_model2 -4.5       5.0   
## negb_model1 -5.0       5.3   
```

```r
pp_check(negb_model1) +
  xlab("sales")
```



Based on the ELPD, there is not much difference in predictive power between the four models. Looking at the simulated datasets for one of the negative binomial models, the results are very similar to the Poisson model from earlier. They share the same problem of overestimating low sales around 8 and underestimating higher sales around 20. The post posterior check for the negative binomial model shows that weekly sales greater than 40 are possible, which was not seen in our observed data.

```r
prediction_summary(model=pois_model1, data=weekly)
```

```
##       mae mae_scaled within_50 within_95
## 1 1.60115  0.5659663 0.6823529 0.9647059
```

```r
prediction_summary(model=pois_model2, data=weekly)
```

```
##       mae mae_scaled within_50 within_95
## 1 1.49365  0.5082933 0.6941176 0.9529412
```

```r
prediction_summary(model=negb_model1, data=weekly)
```

```
##      mae mae_scaled within_50 within_95
## 1 1.7718  0.4343924 0.7529412         1
```

```r
prediction_summary(model=negb_model2, data=weekly)
```

```
##       mae mae_scaled within_50 within_95
## 1 1.47475  0.3941376 0.7529412         1
```

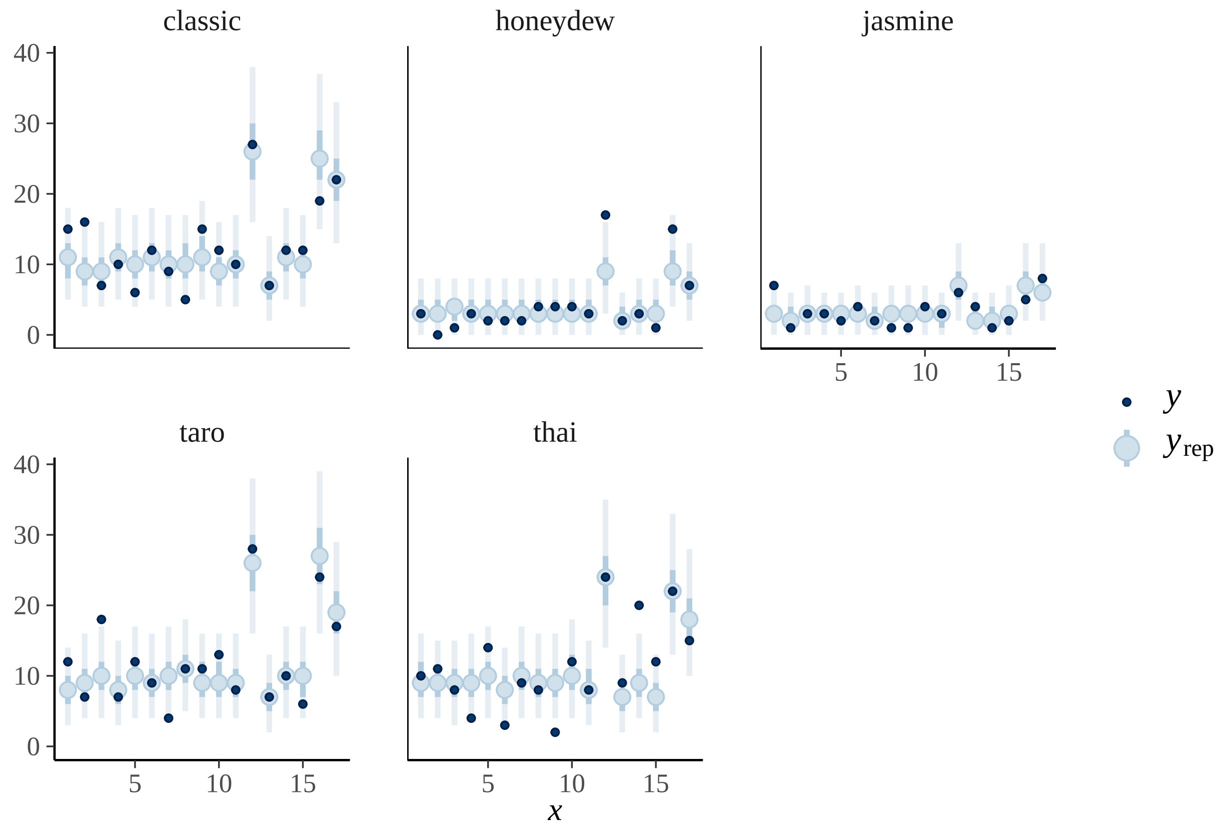
All of the models have a mean absolute error of around 1.5 to 1.75. This tells us that on average, the predicted weekly sales is off from the actual observed value by only about 1.5 to 1.75 sales. We need to keep in mind this is only for data the model has been trained on. Based on the context, an average error of about 1.5 to 1.75 sales per week is not too bad.

Testing our models against the observed data, we see that the lowest errors come from the models without visit_lag1. The mean absolute errors scaled are lower for the negative binomial models, which indicates that on average, the predicted sales from the negative binomial models are fewer standard deviations away from the observed sales than the predicted sales from the Poisson models.

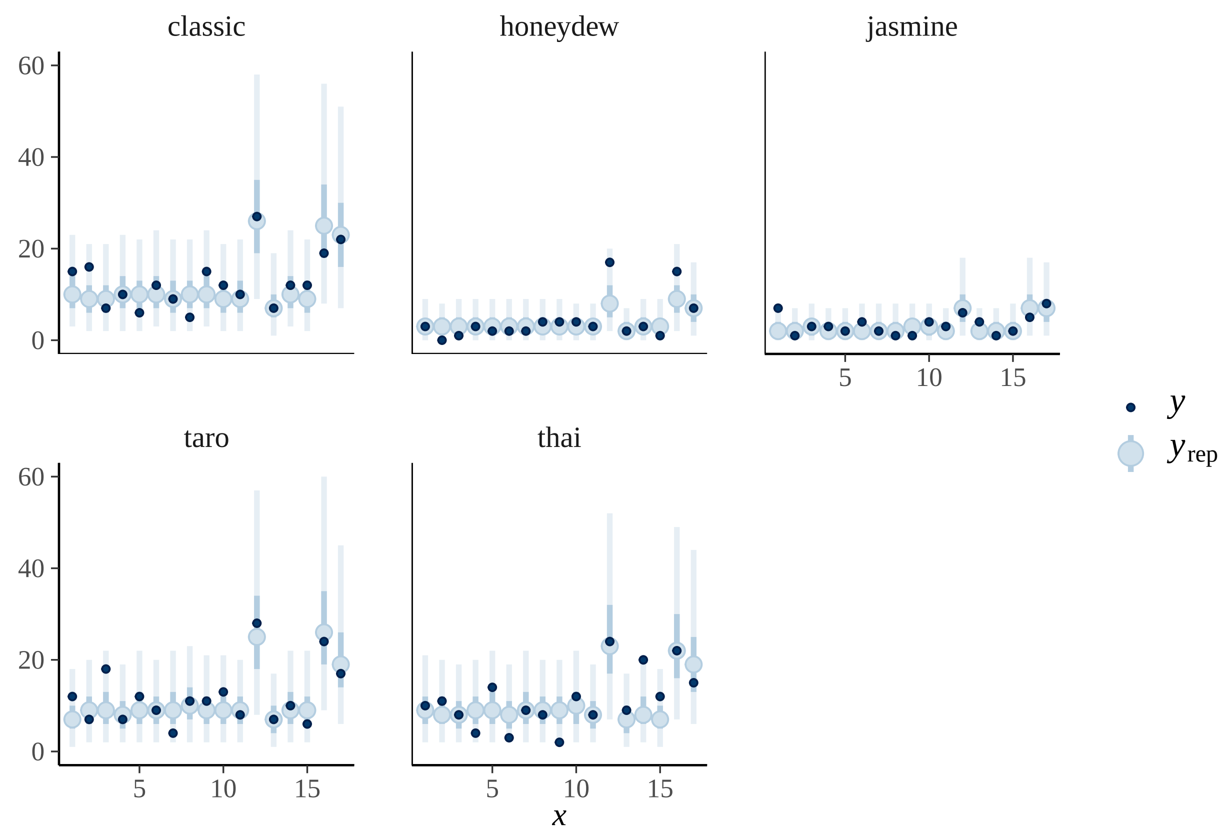
However, this could be attributed to the negative binomial models simply having larger variance than the Poisson models. This is also reflected in the prediction intervals, with the negative binomial models containing a higher percentage of observed values within both their 50% and 95% prediction intervals than the Poisson models.

This is shown in the plots below, where the prediction intervals for the negative binomial model have a bigger range than the intervals for the Poisson model.

```r
set.seed(84735)
exist_predictions_p <- posterior_predict(pois_model2, newdata = weekly)

ppc_intervals_grouped(weekly$sales, yrep = exist_predictions_p,
                      x = as.numeric(weekly$week),
                      group = weekly$tea,
                      prob = 0.5, prob_outer = 0.95,
                      facet_args = list(scales = "fixed"))
```



```r
set.seed(84735)
exist_predictions_nb <- posterior_predict(negb_model2, newdata = weekly)

ppc_intervals_grouped(weekly$sales, yrep = exist_predictions_nb,
                      x = as.numeric(weekly$week),
                      group = weekly$tea,
                      prob = 0.5, prob_outer = 0.95,
                      facet_args = list(scales = "fixed"))
```



The prediction_summary_cv function calculates the cross-validated mean absolute error by training on 4 separate groups and testing on the last group. This is a useful metric for measuring how well our model does at predicting sales for new unseen teas.

```r
new_weekly <- weekly
new_weekly <- weekly %>%
  mutate(tea = factor(tea))
```

```r
set.seed(84735)
prediction_summary_cv(model=pois_model2, data=new_weekly, k=5, group="tea")
```

```
## Warning: There were 3 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.

## Warning: Examine the pairs() plot to diagnose sampling problems
## $folds
##   fold      mae mae_scaled within_50 within_95
## 1    1  1.77220 0.04865899 0.6470588         1
## 2    2  9.09855 0.21401943 0.1764706         1
## 3    3 11.00340 0.11996016 0.2941176         1
## 4    4  3.34985 0.03523209 0.6470588         1
## 5    5  2.32035 0.07656948 0.5294118         1
##
## $cv
##       mae mae_scaled within_50 within_95
## 1 5.50887 0.09888803 0.4588235         1
```

```r
set.seed(84735)
prediction_summary_cv(model=negb_model2, data=new_weekly, k=5, group="tea")
```

```
## Warning: There were 2 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.

## Warning: Examine the pairs() plot to diagnose sampling problems

## $folds
##   fold       mae  mae_scaled within_50 within_95
## 1    1 130.64205 0.009454017 0.7647059         1
## 2    2  20.68195 0.041040138 0.4117647         1
## 3    3  42.44175 0.021293014 0.6470588         1
## 4    4  24.50140 0.027100321 0.8235294         1
## 5    5 499.08040 0.008101829 0.5882353         1
##
## $cv
##        mae mae_scaled within_50 within_95
## 1 143.4695 0.02139786 0.6470588         1
```

Our mean absolute errors are a lot larger than when we trained and tested on the full dataset. This makes sense since we lost an entire tea's worth of observations, and we only had 5 teas. On average, the predicted sales for unseen teas from the Poisson model are about 5.5 sales away from the observed values. The negative binomial model did much worse, with predicted sales being on average about 15 sales away from the observed values.

If we want to know how well our model does at predicting new sales for previously established teas (rather than for new teas), we need to write our own framework.

We can train on the first k rows then test on row k+1 and calculate the mean absolute error. Then, we can train on the first k+1 rows and so on. Let's use this procedure starting with k=14.

```r
train_weekly <- weekly %>%
  filter(week < 14)

test_weekly <- weekly %>%
  filter(week >= 14)
```

```r
forward_chaining_folds <- function(df, time_var) {
  times <- sort(unique(df[[time_var]]))
  folds <- list()

  for (i in 2:length(times)) {
    train_time <- times[1:(i-1)]
    test_time <- times[i]

    train_idx <- which(df[[time_var]] %in% train_time)
    test_idx <- which(df[[time_var]] == test_time)

    folds[[i-1]] <- list(train = train_idx, test = test_idx)
  }

  return(folds)
}

folds <- forward_chaining_folds(test_weekly, "week")
```

```r
set.seed(123)
results <- list()

for (i in seq_along(folds)) {
  fold <- folds[[i]]
  train_data <- bind_rows(train_weekly, test_weekly[fold$train, ])
  test_data <- test_weekly[fold$test, ]

  model <- stan_glmer(
    sales ~ sales_lag1 + holiday + (1 | tea),
    data = train_data, family = poisson,
    prior_intercept = normal(2, .25, autoscale = TRUE),
    prior = normal(0, 2.5, autoscale = TRUE),
    prior_covariance = decov(reg = 1, conc = 1, shape = 1, scale = 1),
    chains = 4, iter = 5000*2, seed = 84735, refresh=0)

  preds <- posterior_predict(model, newdata = test_data)

  # for each column calculate the mean across all posterior draws
  test_data$pred <- apply(preds, 2, mean)

  results[[i]] <- test_data
}
```

```
## Warning: There were 6 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.

## Warning: Examine the pairs() plot to diagnose sampling problems
```

```r
cv_results <- do.call(rbind, results)
```

```r
mean(abs(cv_results$sales - cv_results$pred))
```

```
## [1] 3.588883
```

For the Poisson model, new sales predictions of established teas fall on average 3.6 sales away from their observed values. However note that two out of three weeks in the testing dataset (16 and 17) both contained holidays while the only other week that contained a holiday in the dataset was week 12. Since there was not as much data on holidays in the training data, the predictions for these weeks were not as accurate as they could have been. For future weeks with holidays, we would expect the error to be lower.

Based on these findings, we can see that last week's customers is not a very strong or important predictor of this week's sales. Also, we don't see much of a difference in the posterior datasets between the Poisson and negative binomial models. For simplicity, we can choose pois_model2.

Let Y_{ij} denote the sales for week i and tea type j, and let $X = (X_{ij1}, X_{i2})$ denote last week's type-specific tea sales and whether this week contains a holiday.

Then the hierarchical Poisson model can be written as

$$Y_{ij}|\beta_{0j}, \beta_1, \beta_2 \sim Pois(\lambda_{ij})$$

where

$$\log \lambda_{ij} = \beta_{0j} + \beta_1 X_{ij1} + \beta_2 X_{i2}$$

We can take a closer look at these coefficients.

```r
tidy(pois_model2, effects = "fixed", conf.int = TRUE, conf.level = 0.80)
```

```
## # A tibble: 3 x 5
##   term        estimate std.error conf.low conf.high
##   <chr>          <dbl>     <dbl>    <dbl>     <dbl>
## 1 (Intercept)   1.99     0.205     1.72      2.25  
## 2 sales_lag1   -0.0197   0.00690  -0.0287   -0.0111
## 3 holiday1      0.941    0.0781    0.839     1.04
```

The posterior median relationship for this week's average sales based on last week's sales and whether this week contains a holiday is

$$\log(\lambda_{ij}) = (1.9851 + b_j) - 0.0197 X_{ij1} + 0.9409 X_{i2}$$

Looking at the posterior credible intervals, there is an 80% chance that if last week had no sales and this week did not contain a holiday, the average sales for this week would be between 5.575 and 9.454.

There is an 80% posterior probability that when controlling for tea type and whether this week has a holiday, for every additional milk tea sale made last week, the number of sales this week decreases by between 0.9717 and 0.9889 times.

There is an 80% posterior probability that when controlling for tea type and last week's sales, if this week contains a holiday, the number of tea sales this week increases by between 2.3133 and 2.8261 times.

We can also look at the tea-specific effects.

```
tidy(pois_model2, effects = "ran_vals",
     conf.int = TRUE, conf.level = 0.80) %>%
  select(level, estimate, conf.low, conf.high)
```

```
## # A tibble: 5 x 4
##   level    estimate conf.low conf.high
##   <chr>       <dbl>    <dbl>     <dbl>
## 1 classic     0.542    0.278     0.814
## 2 honeydew   -0.670   -0.964    -0.379
## 3 jasmine    -0.930   -1.23     -0.628
## 4 taro        0.485    0.216     0.754
## 5 thai        0.402    0.134     0.675
```

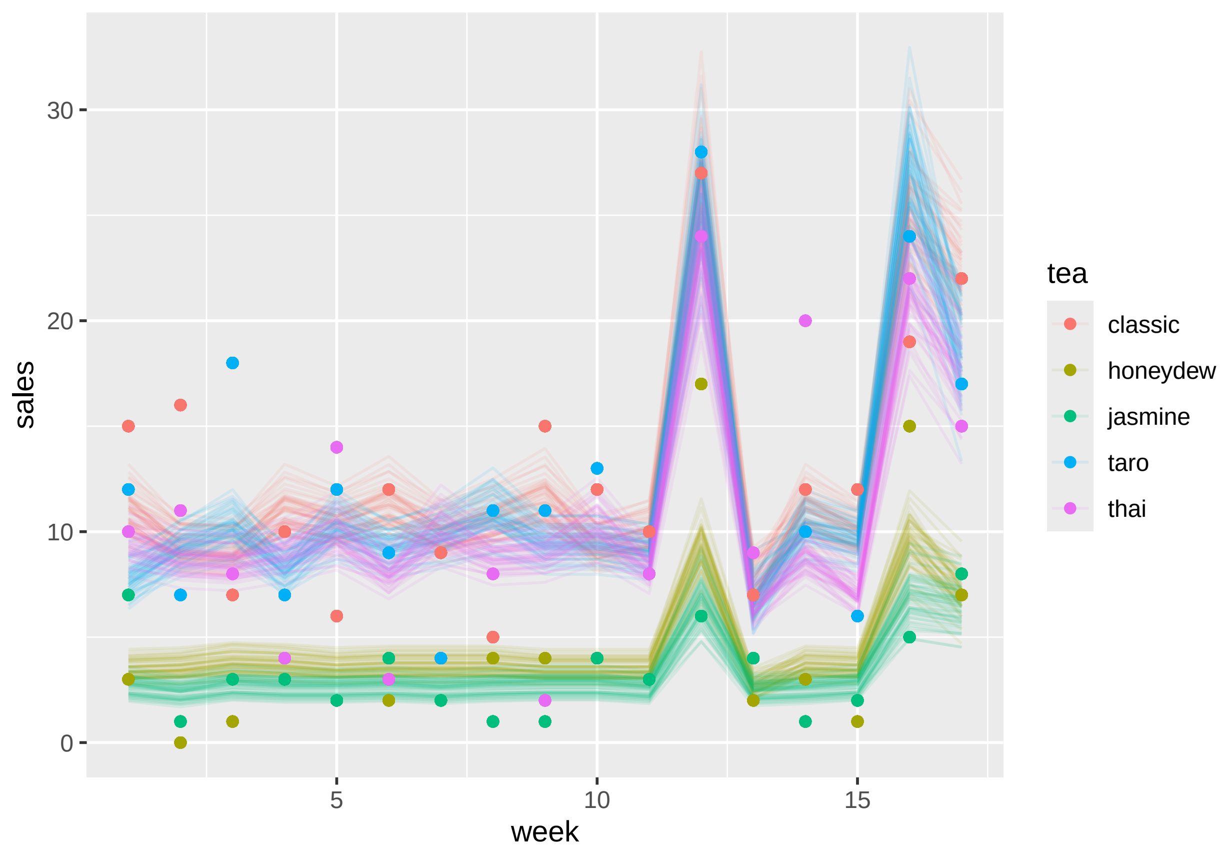
These effects are all relative to a baseline level. We can also interpret these intervals.

There is an 80% chance that classic milk tea sales are between 1.3203 and 2.2577 times higher than the average tea. On the other hand, there is an 80% chance that honeydew milk tea sales are between 0.3813 and 0.6848 times lower than the average tea.

We plot 50 posterior plausible models for average weekly sales using the actual data. The average sales level for weeks without holidays is about the same week to week, with some fluctuations up and down. On holidays, the average sales level jumps for each type of tea.

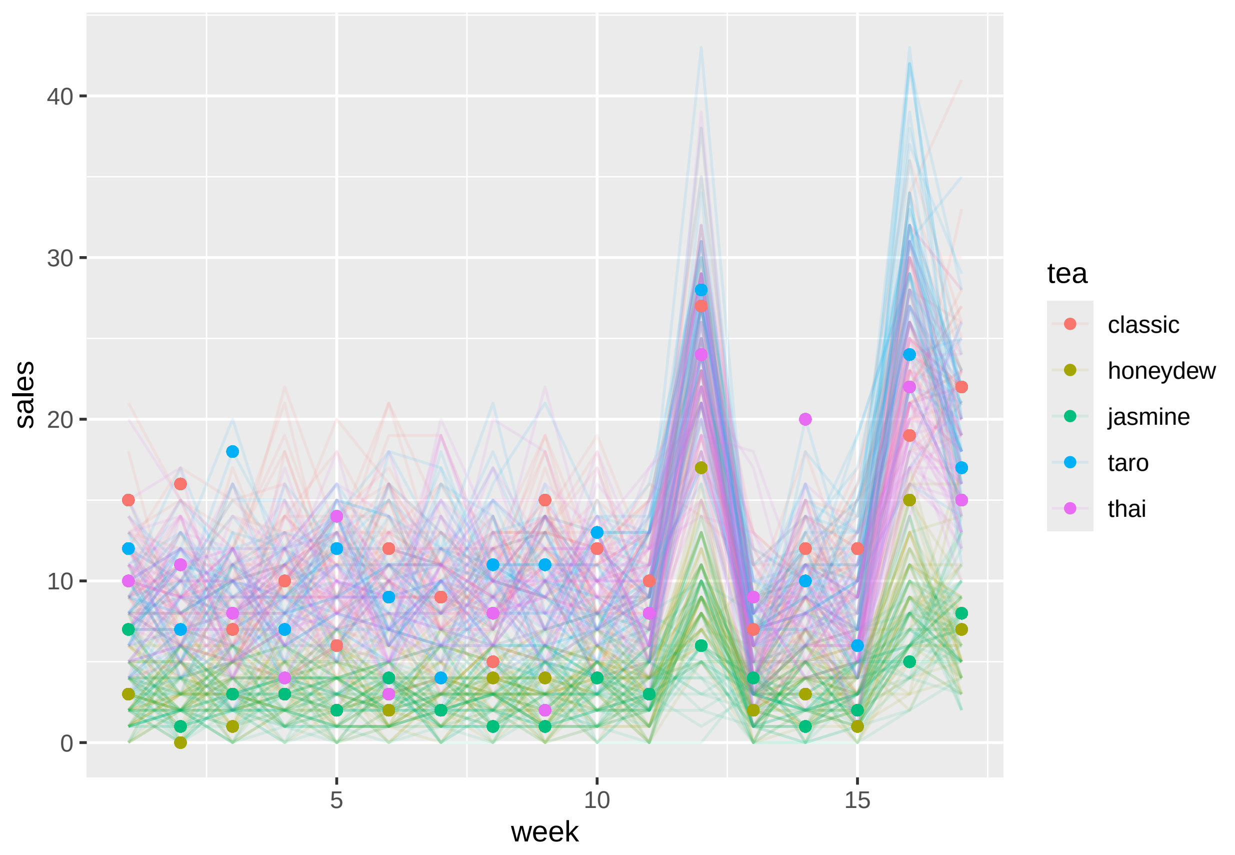
```
weekly %>%
  add_fitted_draws(pois_model2, n = 50) %>%
  ggplot(aes(x = week, y = sales)) +
    geom_line(
      aes(y = .value, group = paste(tea, .draw), color = tea),
      alpha = 0.1) +
    geom_point(aes(color = tea))
```

```
## Warning in fitted_draws.default(model = model, newdata = newdata, ..., n = n): `fitted_draws` and `ad
## - Use [add_]epred_draws() to get the expectation of the posterior predictive.
## - Use [add_]linpred_draws() to get the distribution of the linear predictor.
## - For example, you used [add_]fitted_draws(..., scale = "response"), which
##   means you most likely want [add_]epred_draws(...).
## NOTE: When updating to the new functions, note that the `model` parameter is now
##   named `object` and the `n` parameter is now named `ndraws`.
```



Plotting 50 posterior plausible values for each of our teas shows that the data simulated from our models are not too different from the observed data.

```
weekly %>%
  add_predicted_draws(pois_model2, ndraws = 50) %>%
  ggplot(aes(x = week, y = sales)) +
    geom_line(
      aes(y = .prediction, group = paste(tea, .draw), color = tea),
      alpha = 0.1) +
    geom_point(aes(color = tea))
```

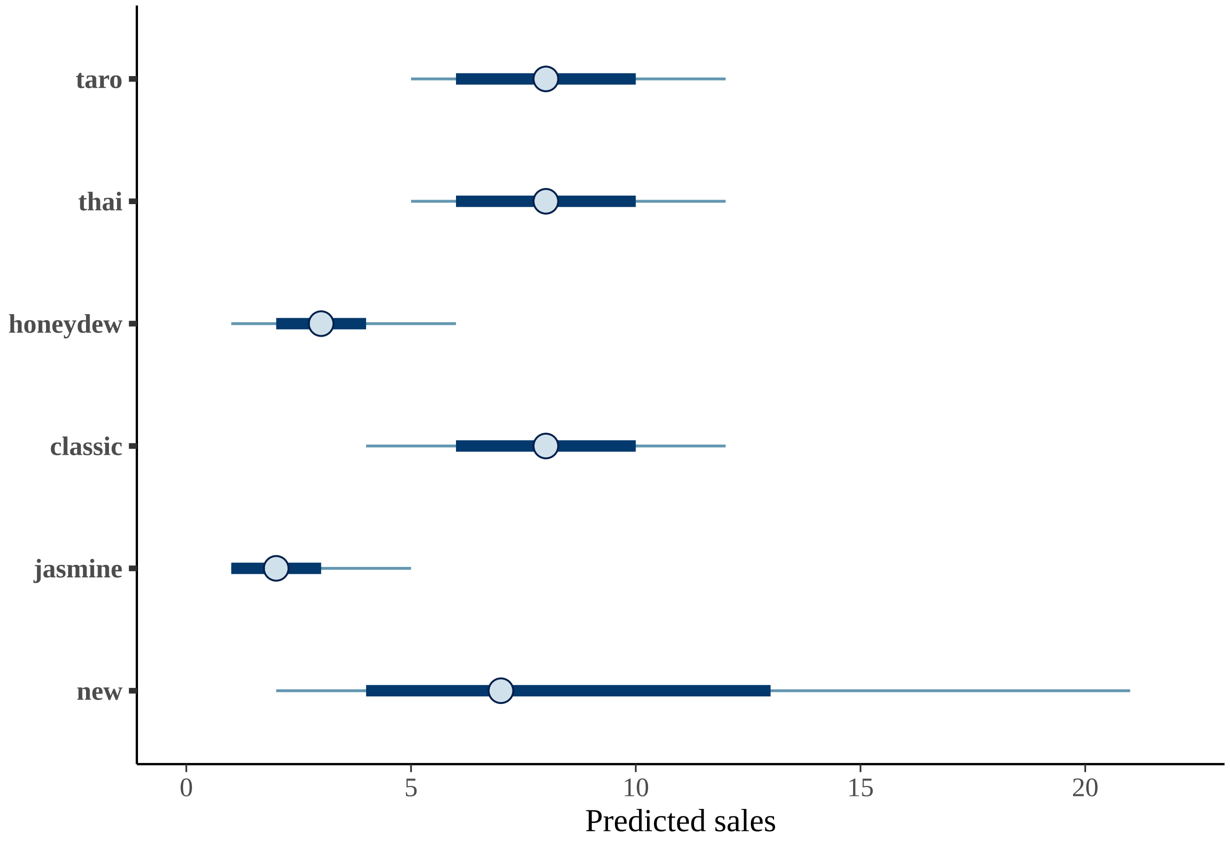


Now we can use our model to predict the sales for next week, from 1/5/2026 to 1/11/2026. We can even use our model to predict the sales for a new milk tea flavor that has never been sold before.

```
next_week = data.frame(
    tea = c("taro", "thai", "honeydew", "classic", "jasmine", "new"),
    sales_lag1 = c(17, 15, 7, 22, 8, 0),
    holiday = as.factor(rep(0, 6)))
next_week
```

```
##        tea sales_lag1 holiday
## 1     taro         17       0
## 2     thai         15       0
## 3 honeydew          7       0
## 4  classic         22       0
## 5  jasmine          8       0
## 6      new          0       0
```

```
set.seed(84735)
predicted_sales <- posterior_predict(
  pois_model2,
  newdata = next_week)
colnames(predicted_sales) <- c(
  "taro", "thai", "honeydew", "classic", "jasmine", "new"
)
mcmc_intervals(
  predicted_sales,
  prob=0.5,
  prob_outer=0.8
) +
  xlab("Predicted sales")
```



We see that different milk teas have different lengths for their prediction intervals. The lengths of these intervals reflect the uncertainty of our predictions. For example, classic milk tea sales tend to vary more week to week compared to jasmine milk tea sales, hence the longer interval. We also have much more data on the five previously established teas than the new tea, so the interval for the new tea is much larger than any of the other intervals.

There is a 50% chance that next week's sales for the new tea is between 4 and 13. There is an 80% chance that next week's sales for the new tea is between 2 and 21.