DATA 237: Visualizing Model Outputs

Alex Kale

2025-02-25

Very brief introduction

Purpose of today's lesson is to get some practical perspective on how to create uncertainty visualizations, and how to visualize outputs from statistical models.

Visualization for model interpretability is one of the most important use cases for visualization in data science, yet it is seldom emphasized in courses like this one.

We expect students to know some regression, but we'll explain what the syntax is doing as we go.

We will use visualization tools built on `ggplot2` and `brms` (Bayesian regression models). The reasons for using these tools are that:

1. `ggplot2` is an excellent implementation of grammar of graphics, which has been extended with `ggdist` to support unparalleled flexibility in uncertainty vis.
2. `brms` gives us sample-based representations of uncertainty from models, which makes it possible to quantify uncertainty and create the widest possible range of uncertainty visualizations.

Loading the prepping the data

We're going to use data about student absences for today's demo.

```
df = read_csv("../data/students.csv", show_col_types = FALSE)
head(df)
```

```
## # A tibble: 6 x 10
##     age address travel_time study_time failures internet absences g_edu g_job
##   <dbl> <chr>         <dbl>      <dbl>    <dbl> <chr>       <dbl> <dbl> <chr>
## 1    18 urban         27.2        3.03        0 no              6     4 at_home
## 2    17 urban         11.0        4.15        0 yes             4     1 other
## 3    15 urban          6.57       2.02        3 yes            10     1 at_home
## 4    15 urban          9.98       6.47        0 yes             2     4 health
## 5    16 urban         12.0        4.32        0 no              4     3 other
## 6    16 urban         14.3        3.11        0 yes            10     4 services
## # i 1 more variable: alcohol <dbl>
```

Change anything we need to change before modeling. Mostly casting discrete variables as factors, so the model uses dummy variables for them. Also, centering continuous variables.

```
model_df = df |> mutate(
  # factors
  address = as.factor(address),
  failures = as.factor(failures),
  internet = as.factor(internet),
  g_edu = as.factor(g_edu),
  g_job = as.factor(g_job),
  # centered continuous predictors
```

```
  c_age = age - mean(age),
  c_tt = travel_time - mean(travel_time),
  c_st = study_time - mean(study_time),
  c_alc = alcohol - mean(alcohol),
  # fake y axis var
  y = 0
)
```

Choosing a model family

First a normal model.

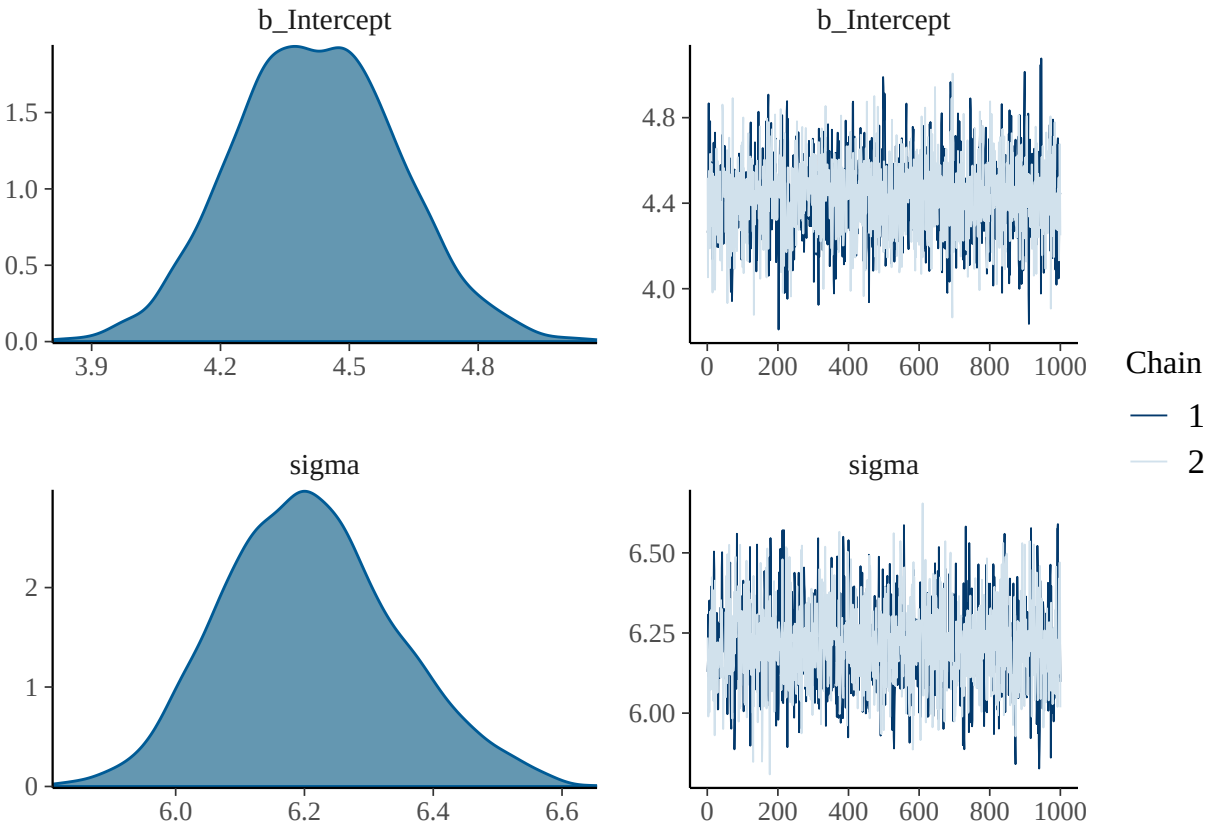
```
m_norm = brm(
  bf("absences ~ 1"),
  family = "normal",
  data = model_df,
  iter = 2000, warmup = 1000, chains = 2,
  file = "../data/models/m0.rds")
```

Typical diagnostics. (Explain what I'm looking for: Rhat, ESS, trace mixing, multicolinearity)

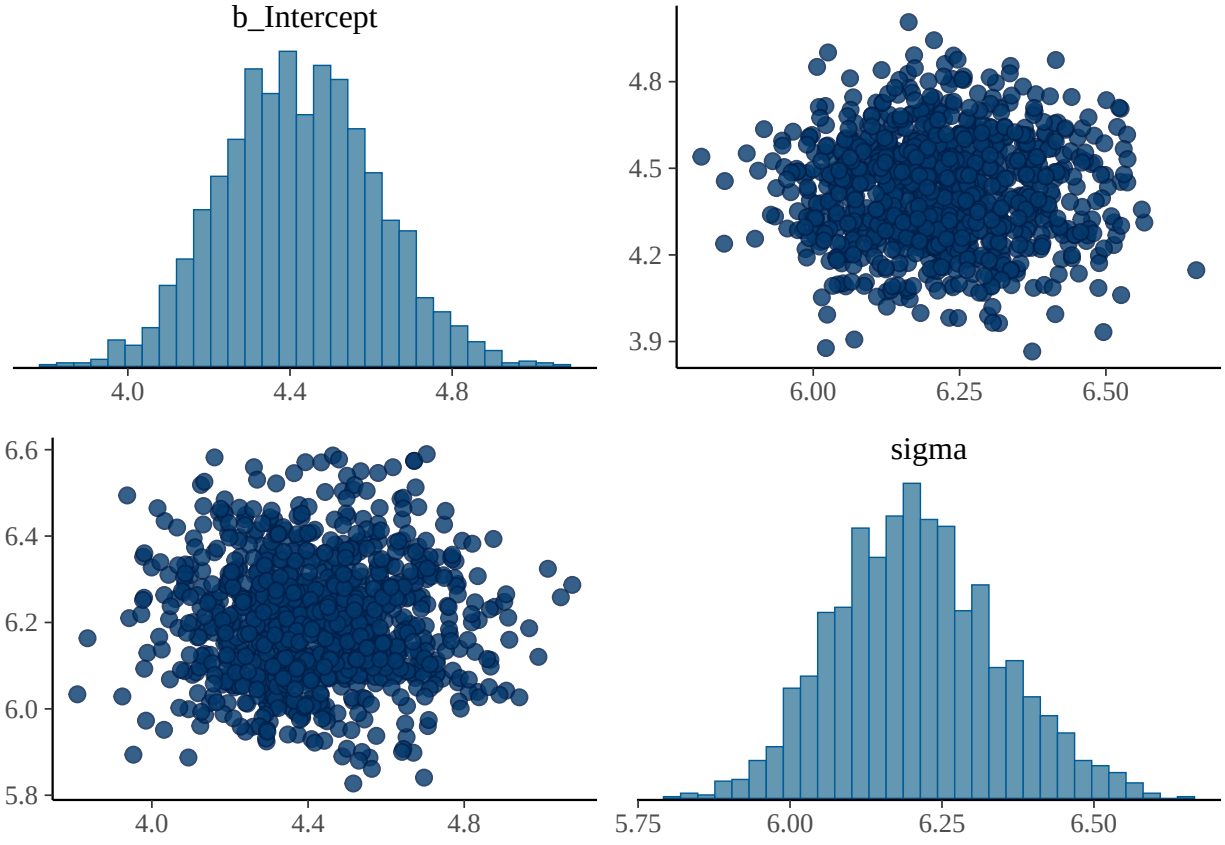
```
summary(m_norm)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: absences ~ 1
##    Data: model_df (Number of observations: 1044)
##   Draws: 2 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 2000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     4.42      0.19     4.07     4.81 1.00     1772     1489
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     6.21      0.13     5.96     6.49 1.00     1763     1372
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
plot(m_norm)
```

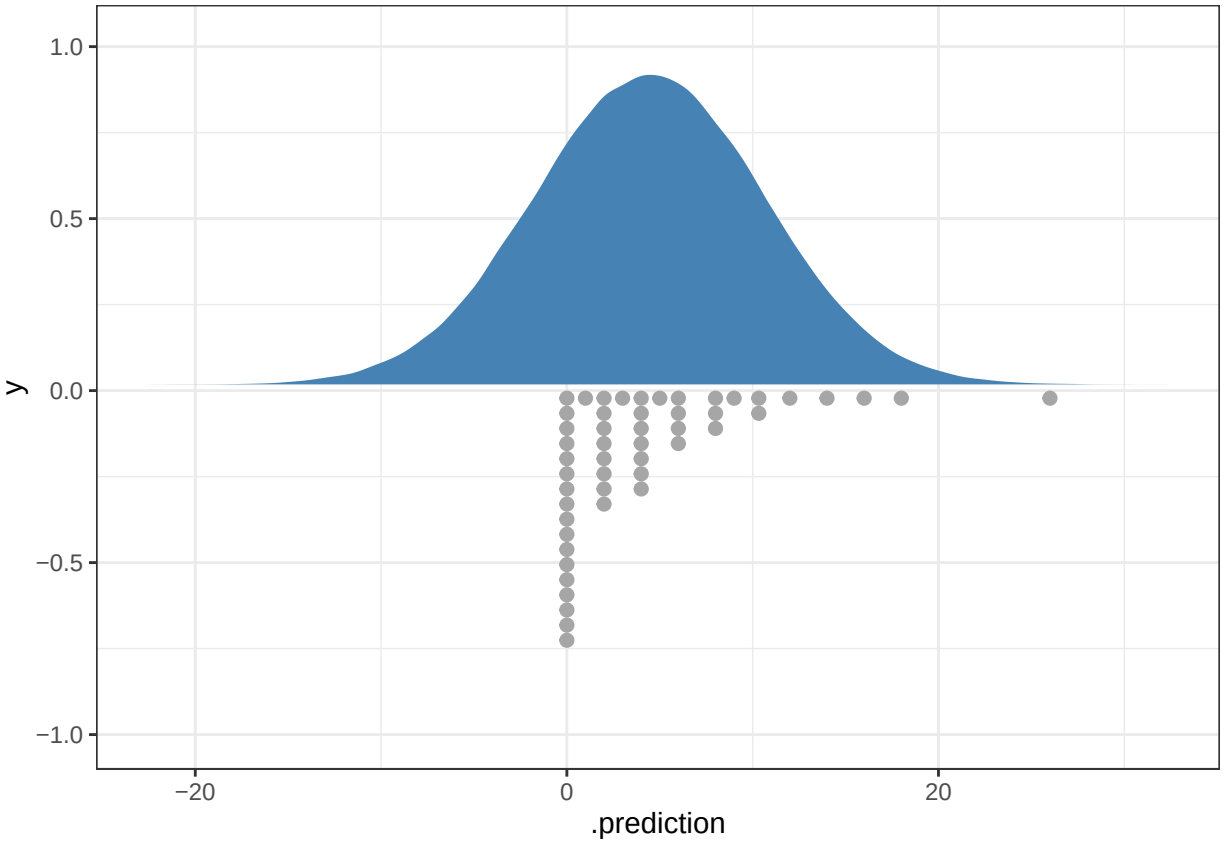

b_Intercept
b_Intercept
sigma
sigma
Chain
1
2

```
pairs(m_norm)
```


b_Intercept
sigma

Posterior predictive check. (Lower bound of zero is an issue)

```
model_df |>
  select(absences, y) |>
  add_predicted_draws(m_norm, ndraws = 200) |>
  ggplot(aes(x = .prediction, y = y)) +
  stat_slab(justification = -0.02, fill = "steelblue") +
  stat_dots(aes(x = absences), quantiles = 50, side = "bottom", scale = 0.75, data = model_df) +
  theme_bw()
```



Now a lognormal model (actually need a hurdle model to handle the zeros).

```
# show first with lognormal (fill give an error)
# talk briefly about how we can handle the zeros: log(y+1), 0 => 0.0001, filter(absences > 0)
m_hlogn = brm(
  bf("absences ~ 1"),
  family = hurdle_lognormal(),
  data = model_df,
  iter = 2000, warmup = 1000, chains = 2,
  file = "../data/models/m1.rds")
```

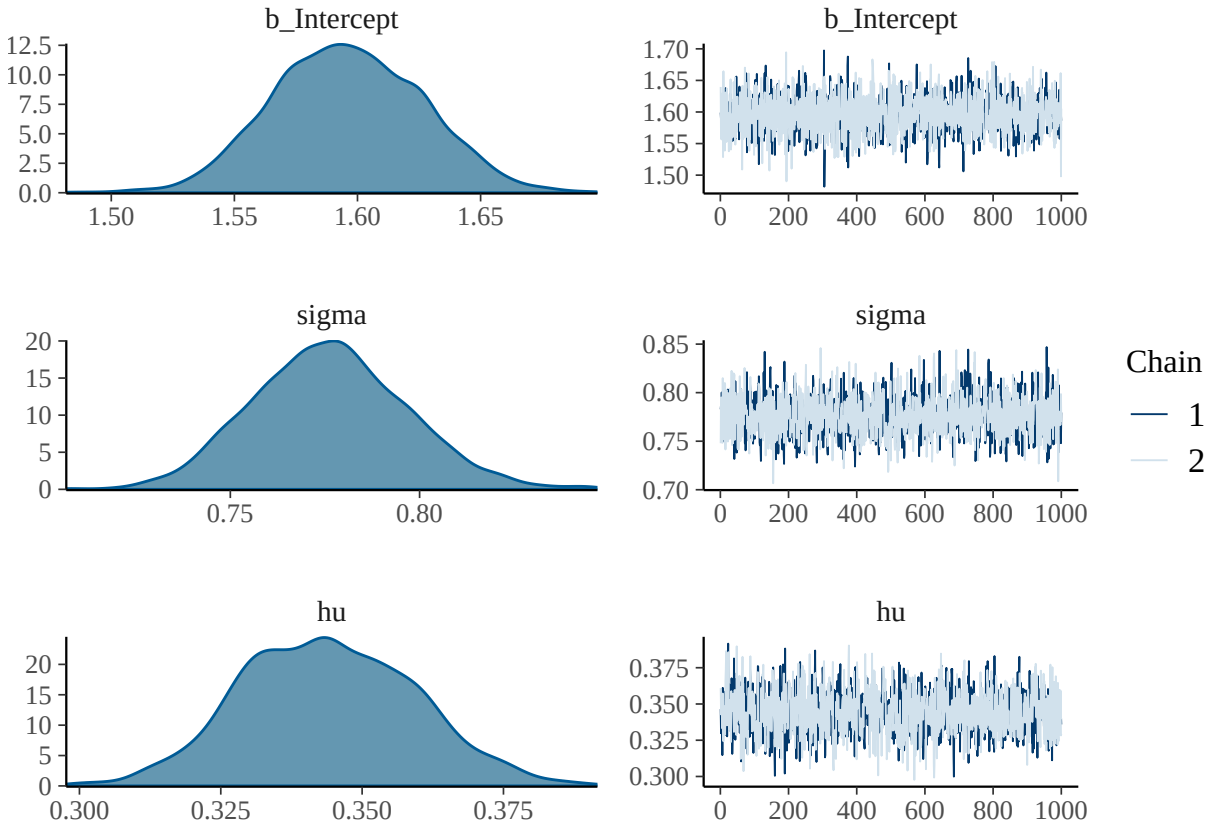
Diagnostics

```
summary(m_hlogn)
```

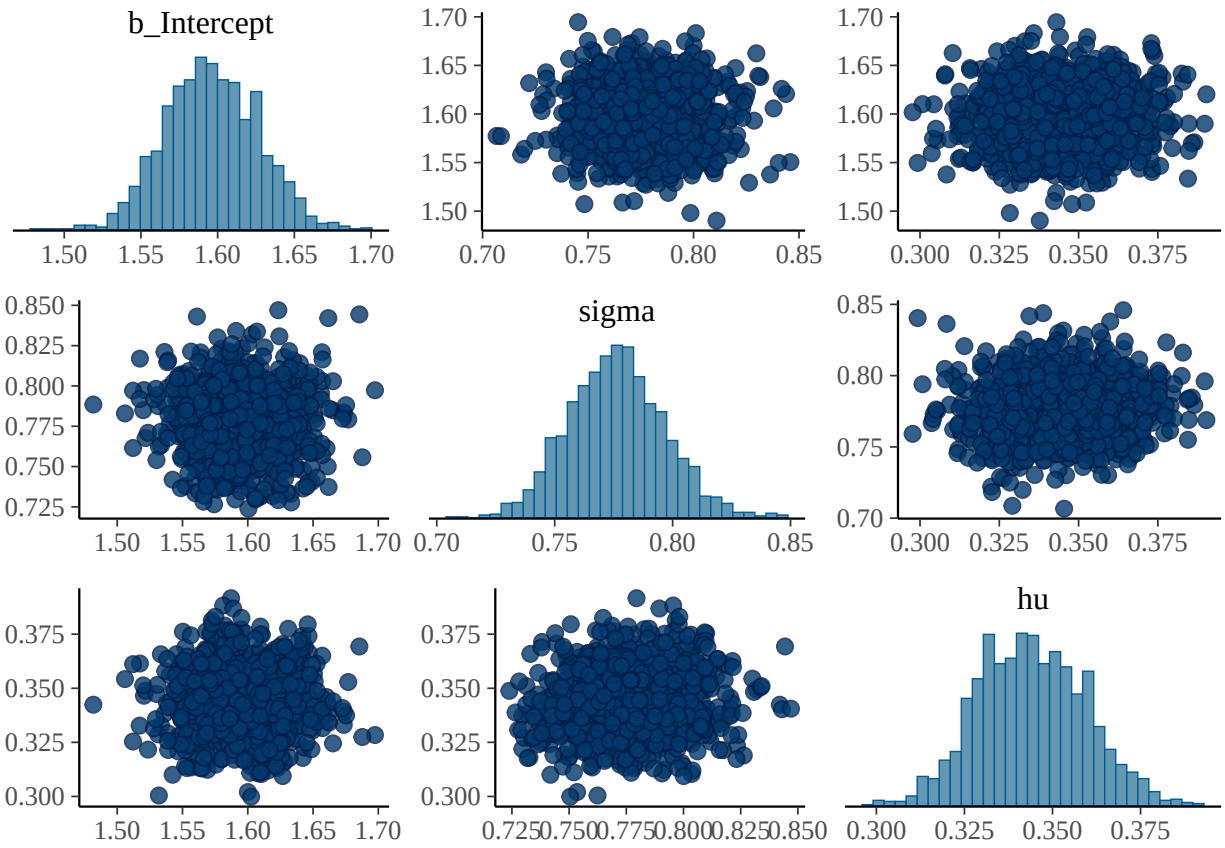
```
##  Family: hurdle_lognormal
##   Links: mu = identity; sigma = identity; hu = identity
## Formula: absences ~ 1
##    Data: model_df (Number of observations: 1044)
```

```
##   Draws: 2 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 2000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     1.60      0.03     1.54     1.65 1.00     2834     1498
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.78      0.02     0.74     0.82 1.00     1751     1547
## hu        0.34      0.02     0.31     0.37 1.00     1615     1585
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
plot(m_hlogn)
```

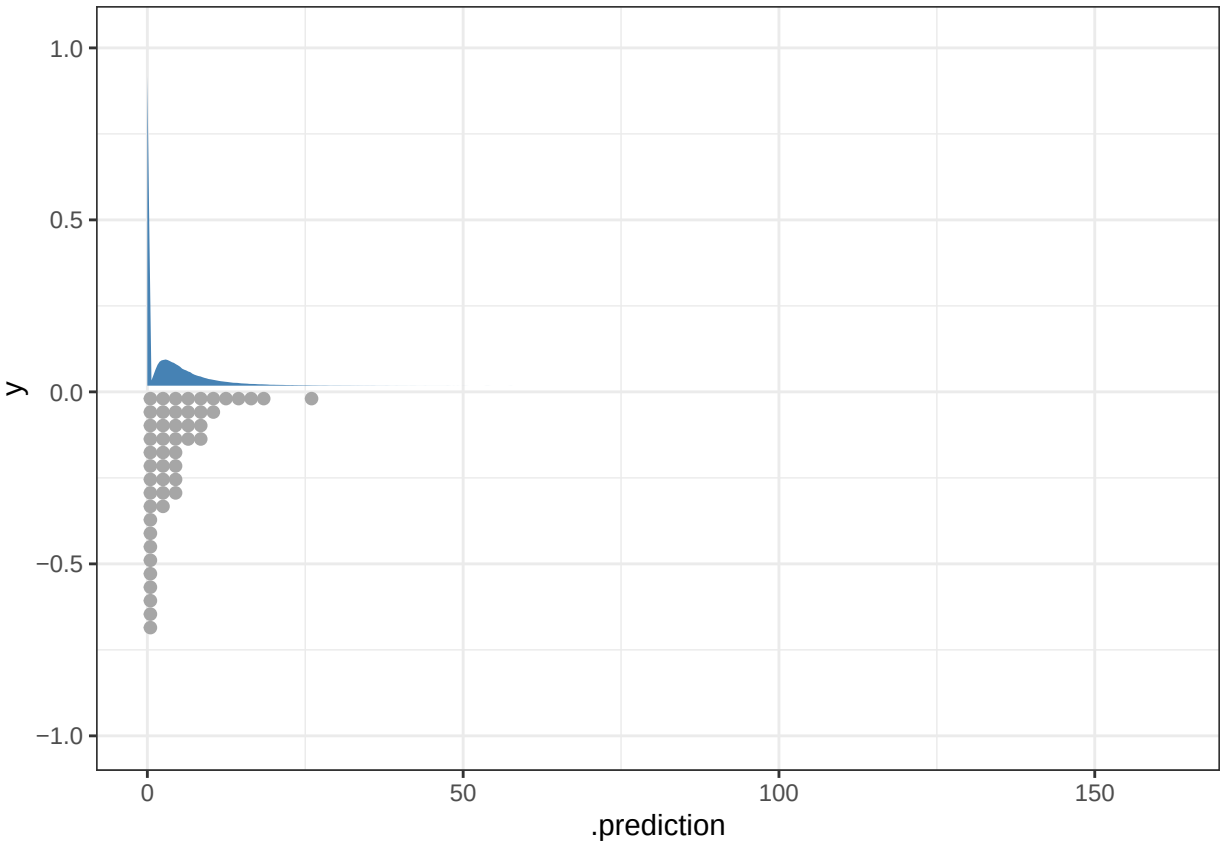



```
pairs(m_hlogn)
```



PP check. Point out improvement and where we still have room to do better.

```
model_df |>
  select(absences, y) |>
  add_predicted_draws(m_hlogn, ndraws = 200) |>
  ggplot(aes(x = .prediction, y = y)) +
  stat_slab(justification = -0.02, fill = "steelblue") +
  stat_dots(aes(x = absences), quantiles = 50, side = "bottom", scale = 0.75, data = model_df) +
  theme_bw()
```



Choosing a set of predictors

Exploratory vis in this section:

We would usually rely on visualization to identify predictors of interest. There are a few ways we can do this. One way would be typical exploratory data analysis with `ggplot2`. Another way would be to create posterior predictive checks with the previous model, conditioning the relationships learned by the model on possible predictors of interest that have not yet been included in the model. The key idea here is that we want to find structure in the data that our previous model migtht not have accounted for.

For today's demo, I've chosen three predictors of interest from the table below: age, study time, and guardian's level of education.

```
head(model_df)
```

```
## # A tibble: 6 x 15
##     age address travel_time study_time failures internet absences g_edu g_job
##   <dbl> <fct>         <dbl>      <dbl> <fct>    <fct>       <dbl> <fct> <fct>
## 1    18 urban         27.2        3.03 0        no              6 4     at_home
## 2    17 urban         11.0        4.15 0        yes             4 1     other
## 3    15 urban          6.57       2.02 3        yes            10 1     at_home
## 4    15 urban          9.98       6.47 0        yes             2 4     health
## 5    16 urban         12.0        4.32 0        no              4 3     other
## 6    16 urban         14.3        3.11 0        yes            10 4     services
## # i 6 more variables: alcohol <dbl>, c_age <dbl>, c_tt <dbl>, c_st <dbl>,
## #   c_alc <dbl>, y <dbl>
```

Incorporating predictors

Add predictors into a model. No interactions in this example, but model checks below show we should add them. Point out that I would usually add predictors to a series of models one-by-one (a model expansion workflow).

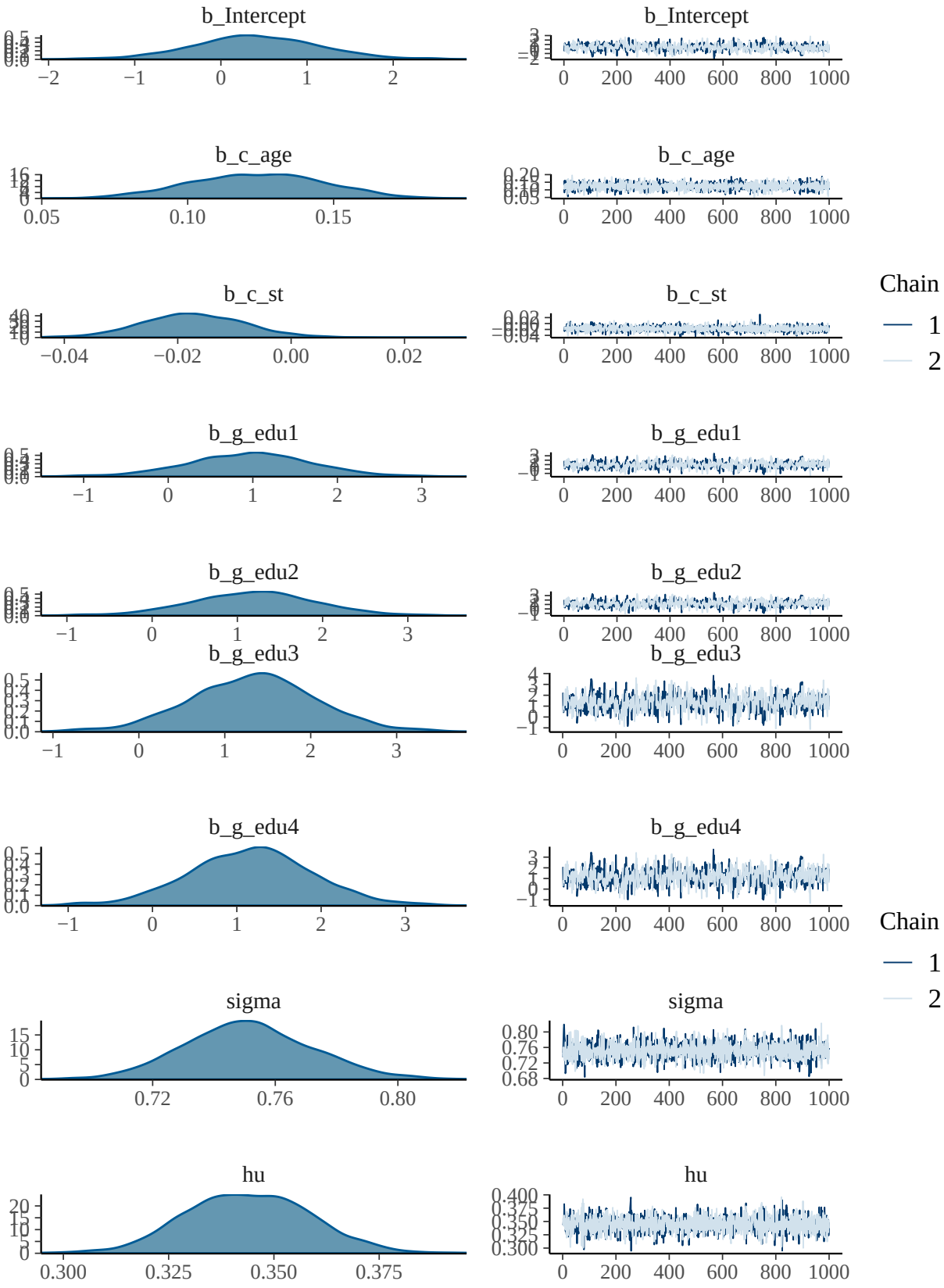
```
m_mains = brm(
  bf("absences ~ c_age + c_st + g_edu"),
  family = hurdle_lognormal(),
  data = model_df,
  iter = 2000, warmup = 1000, chains = 2,
  file = "../data/models/m2.rds")
```

Diagnostics.

```
summary(m_mains)
```

```
##  Family: hurdle_lognormal
##   Links: mu = identity; sigma = identity; hu = identity
## Formula: absences ~ c_age + c_st + g_edu
##    Data: model_df (Number of observations: 1044)
##   Draws: 2 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 2000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     0.41      0.74    -1.02     1.87 1.00      587      858
## c_age         0.12      0.02     0.08     0.17 1.00     2290     1374
## c_st         -0.02      0.01    -0.03     0.00 1.00     2632     1416
## g_edu1        0.97      0.74    -0.50     2.41 1.00      581      789
## g_edu2        1.19      0.74    -0.30     2.61 1.00      591      834
## g_edu3        1.33      0.74    -0.13     2.76 1.00      598      792
## g_edu4        1.17      0.74    -0.29     2.61 1.00      596      873
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.75      0.02     0.71     0.79 1.00     2340     1342
## hu        0.34      0.01     0.32     0.37 1.00     2605     1310
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

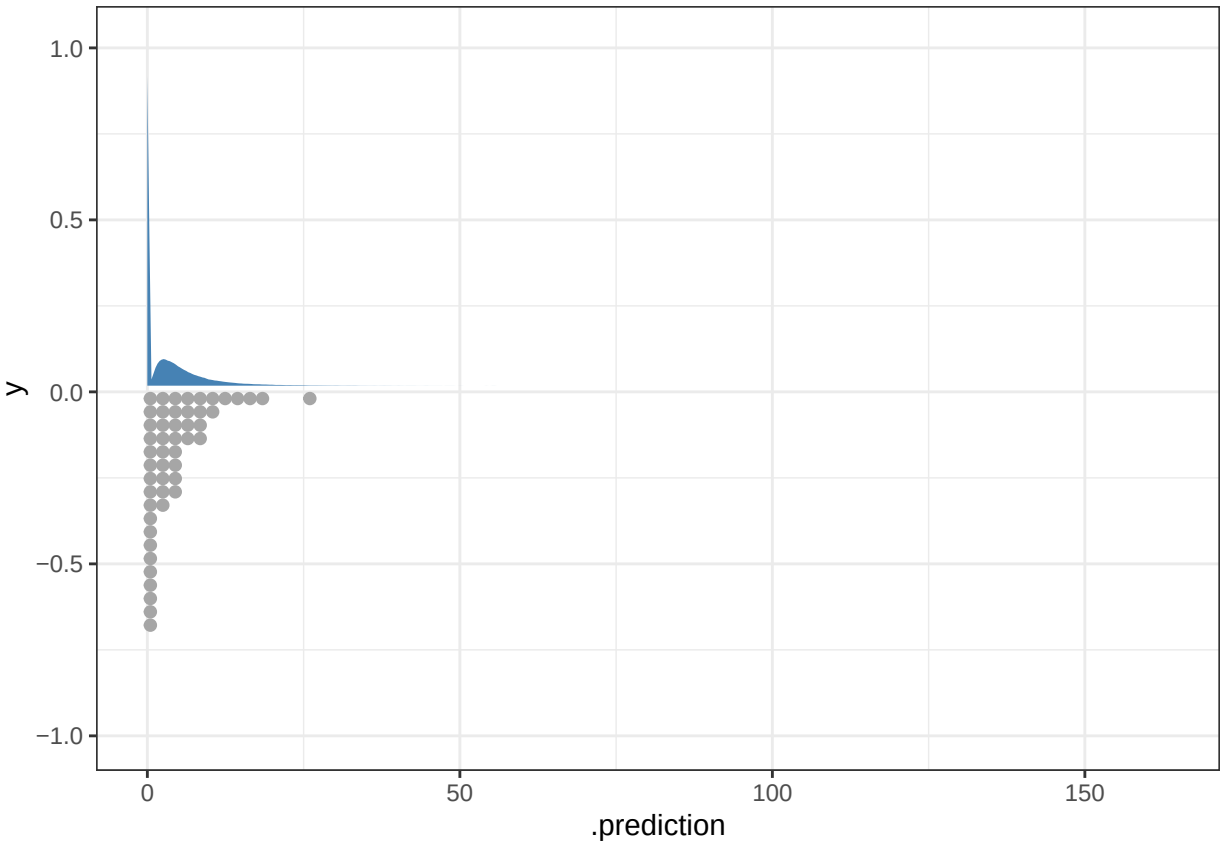
```
plot(m_mains)
```




```
pairs(m_mains) # these scale poorly to larger models
```

PP checks borrowing code from above, now with new model.

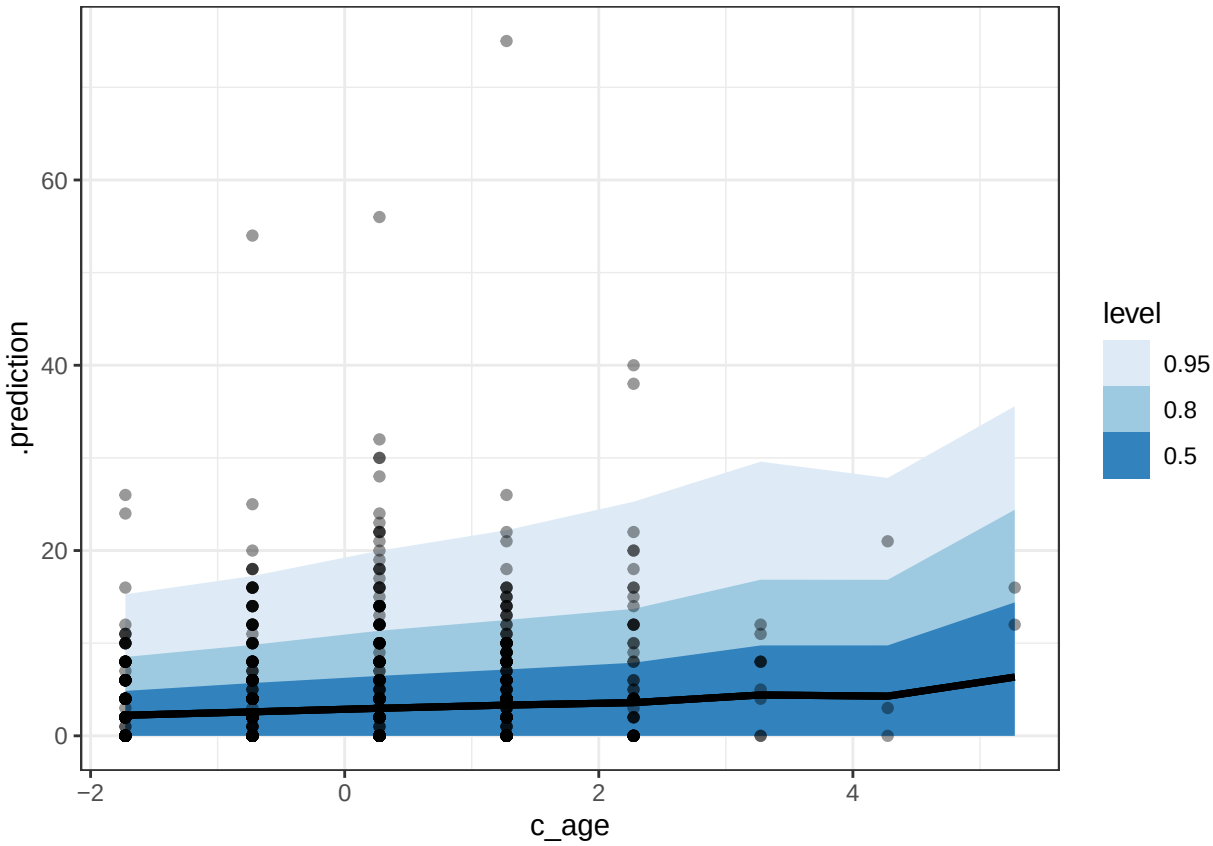
```
model_df |>
  select(absences, c_age, c_st, g_edu) |>
  mutate(y = 0) |>
  add_predicted_draws(m_mains, ndraws = 200) |>
  ggplot(aes(x = .prediction, y = y)) +
  stat_slab(justification = -0.02, fill = "steelblue") +
  stat_dots(aes(x = absences), quantiles = 50, side = "bottom", scale = 0.75, data = model_df) +
  theme_bw()
```



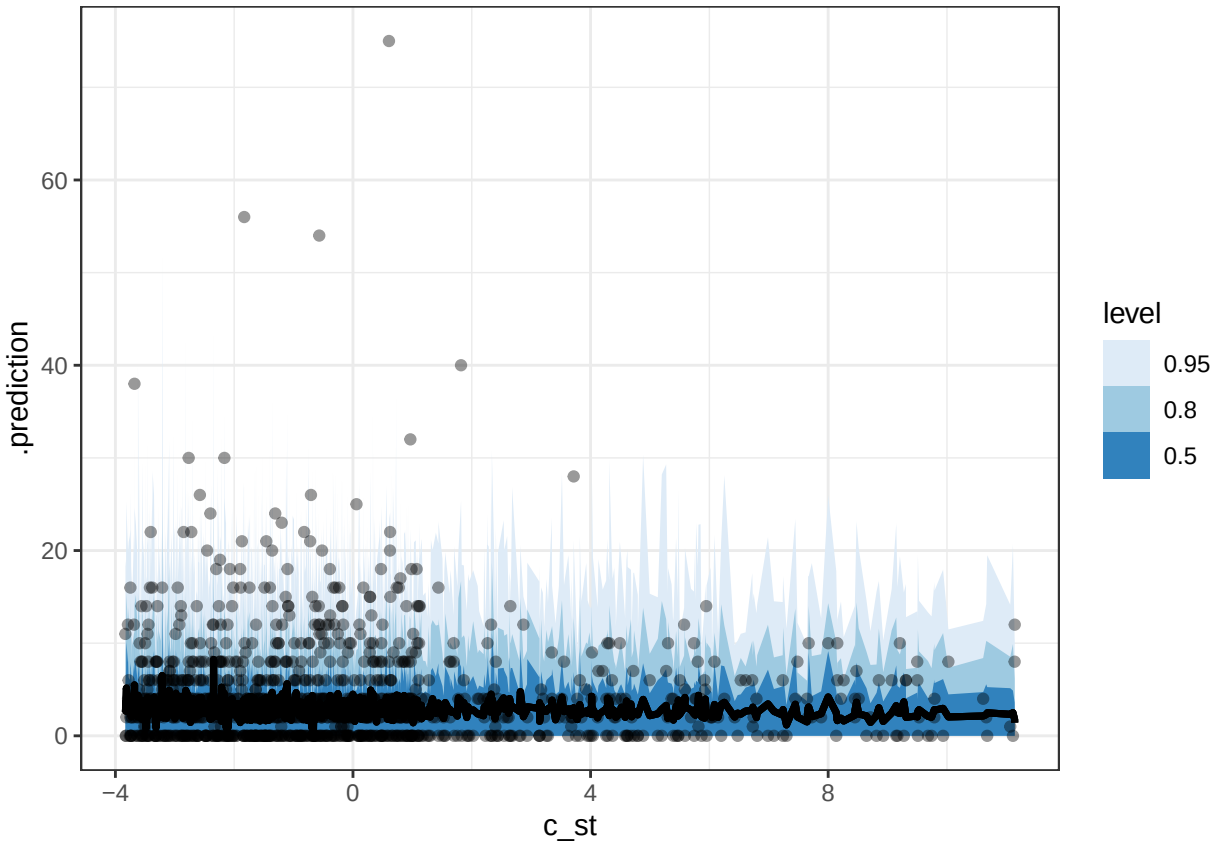
Additional predictive checks conditioning on predictors

This is what we do to check whether we've accounted for important structure in our data.

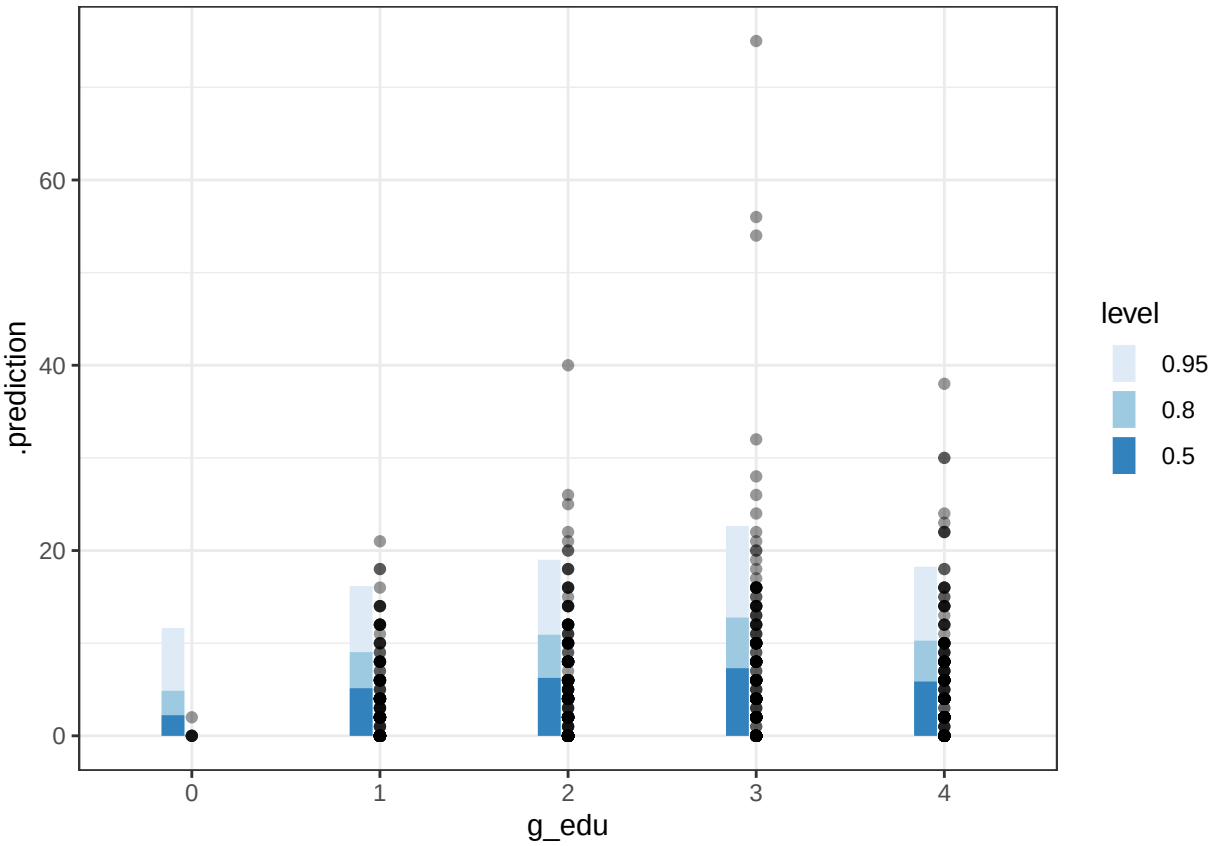
```
# need a new form of model check visualization
model_df |>
  select(absences, c_age, c_st, g_edu) |>
  add_predicted_draws(m_mains, ndraws = 200) |>
  ggplot(aes(x = c_age, y = .prediction)) +
  stat_lineribbon(.width = c(.50, .80, .95)) +
  scale_fill_brewer() +
  geom_point(aes(y = absences), alpha = 0.4, data = model_df) +
  theme_bw()
```




```
# repeat mc for next predictor
model_df |>
  select(absences, c_age, c_st, g_edu) |>
  add_predicted_draws(m_mains, ndraws = 200) |>
  ggplot(aes(x = c_st, y = .prediction)) +
  stat_lineribbon(.width = c(.50, .80, .95)) +
  scale_fill_brewer() +
  geom_point(aes(y = absences), alpha = 0.4, data = model_df) +
  theme_bw()
```



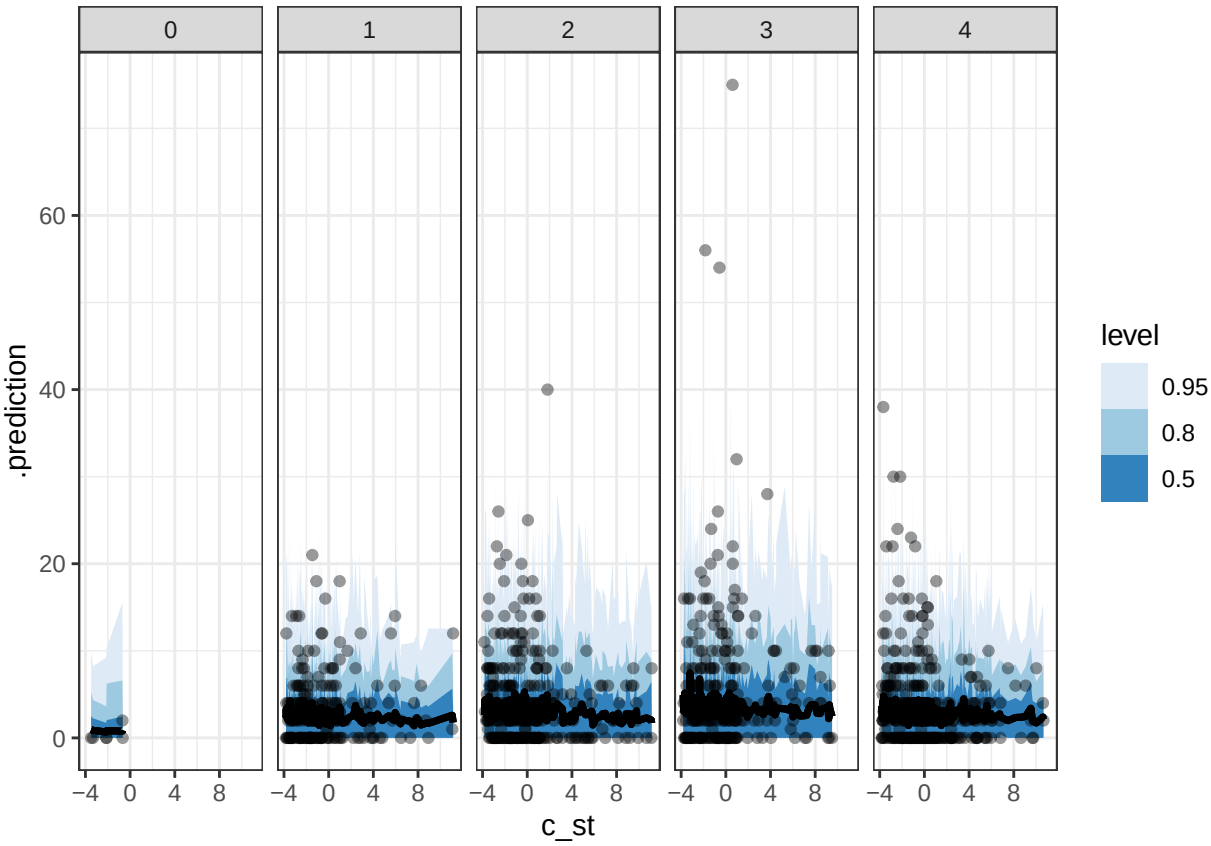

```
# modify by data type of predictor
model_df |>
  select(absences, c_age, c_st, g_edu) |>
  add_predicted_draws(m_mains, ndraws = 200) |>
  ggplot(aes(x = g_edu, y = .prediction)) +
  stat_interval(.width = c(.50, .80, .95), position = position_nudge(x = -0.1)) +
  scale_color_brewer() +
  geom_point(aes(y = absences), alpha = 0.4, data = model_df) +
  theme_bw()
```



I can sort of see where these points I'm not predicting well end up in across charts, and this gives me a sense of how to modify my model.

If we condition on two predictors at once, we can see where we might need to include interaction effects in the model, e.g., `absences ~ c_age * c_st * g_edu`.

```
# extend mc to show unmodeled potential interaction
model_df |>
  select(absences, c_age, c_st, g_edu) |>
  add_predicted_draws(m_mains, ndraws = 200) |>
  ggplot(aes(x = c_st, y = .prediction)) +
  stat_lineribbon(.width = c(.50, .80, .95)) +
  scale_fill_brewer() +
  geom_point(aes(y = absences), alpha = 0.4, data = model_df) +
  theme_bw() +
  facet_grid(. ~ g_edu)
```



Visualizing inferential uncertainty

Now that we have predictors in our model, we want ways to summarize the relationship between predictors and the outcome variable. For example, this is the kind of inference we typically capture in a t-test, confidence interval, or trend line. Our approach here emphasizes the visualization of uncertainty about the model's parameters.

You can find some nice examples of inferential uncertainty visualization with `ggdist` here: http://mjskay.github.io/tidybayes/articles/tidy-brms.html#posterior-means-and-predictions

One critical thing we're going to do below is show the *expected value of predictions*, which is distinct from showing the predictions themselves. To do this, we will rely on the `tidybayes` function `add_epred_draws` rather than `add_predicted_draws`, which we used above. You can find an explanation of this syntax here: https://mjskay.github.io/tidybayes/reference/add_predicted_draws.html

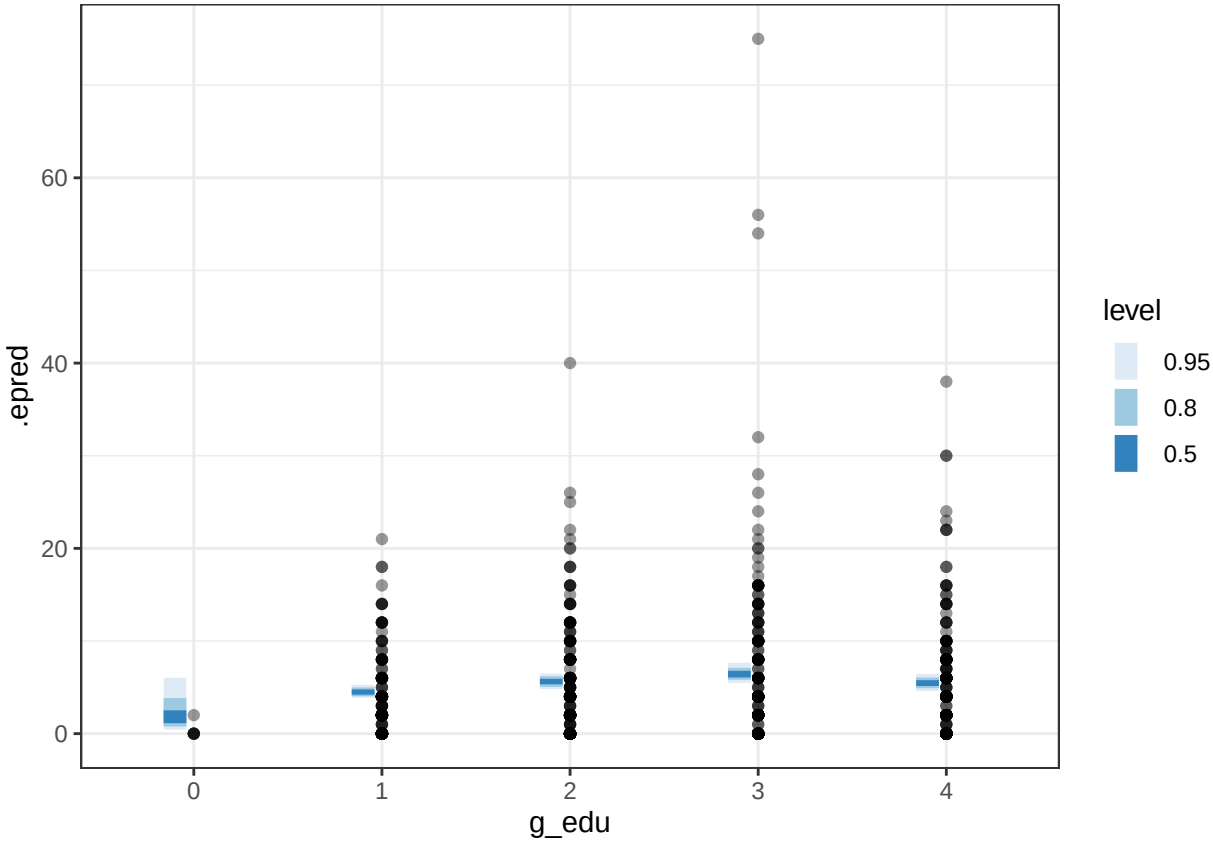
```
# precompute a data grid for next few vises
# this is a very large obj to hold in memory, so it's best to run this code only once
# using `data_grid` instead of `select` now, which crosses all values of each predictor
predictor_grid = model_df |>
  data_grid(c_age, c_st, g_edu)
```

Effect of guardian education.

Now we need to *marginalize* by averaging the effect we are interested in looking at over the conditions in the model that we do not want to compare. We do this for inferential uncertainty to get a summary of the expected/average impact of a predictor, whereas for predictive uncertainty, we wanted to see all the variation in possible outcomes that the model had learned. See here for a description and examples of marginalization: https://htmlpreview.github.io/?https://github.com/mjskay/uncertainty-examples/blob/master/marginal-effects_categorical-predictor.html

```
# modifying our mc to show inferential uncertainty about central tendency
predictor_grid |>
  add_epred_draws(m_mains, ndraws = 200) |>
  group_by(g_edu, .draw) |>              # marginalization
  summarise(.epred = mean(.epred)) |>    # marginalization
  ggplot(aes(x = g_edu, y = .epred)) +
  stat_interval(.width = c(.50, .80, .95), position = position_nudge(x = -0.1)) +
  scale_color_brewer() +
  geom_point(aes(y = absences), alpha = 0.4, data = model_df) +
  theme_bw()
```

```
## `summarise()` has grouped output by 'g_edu'. You can override using the
## `.groups` argument.
```



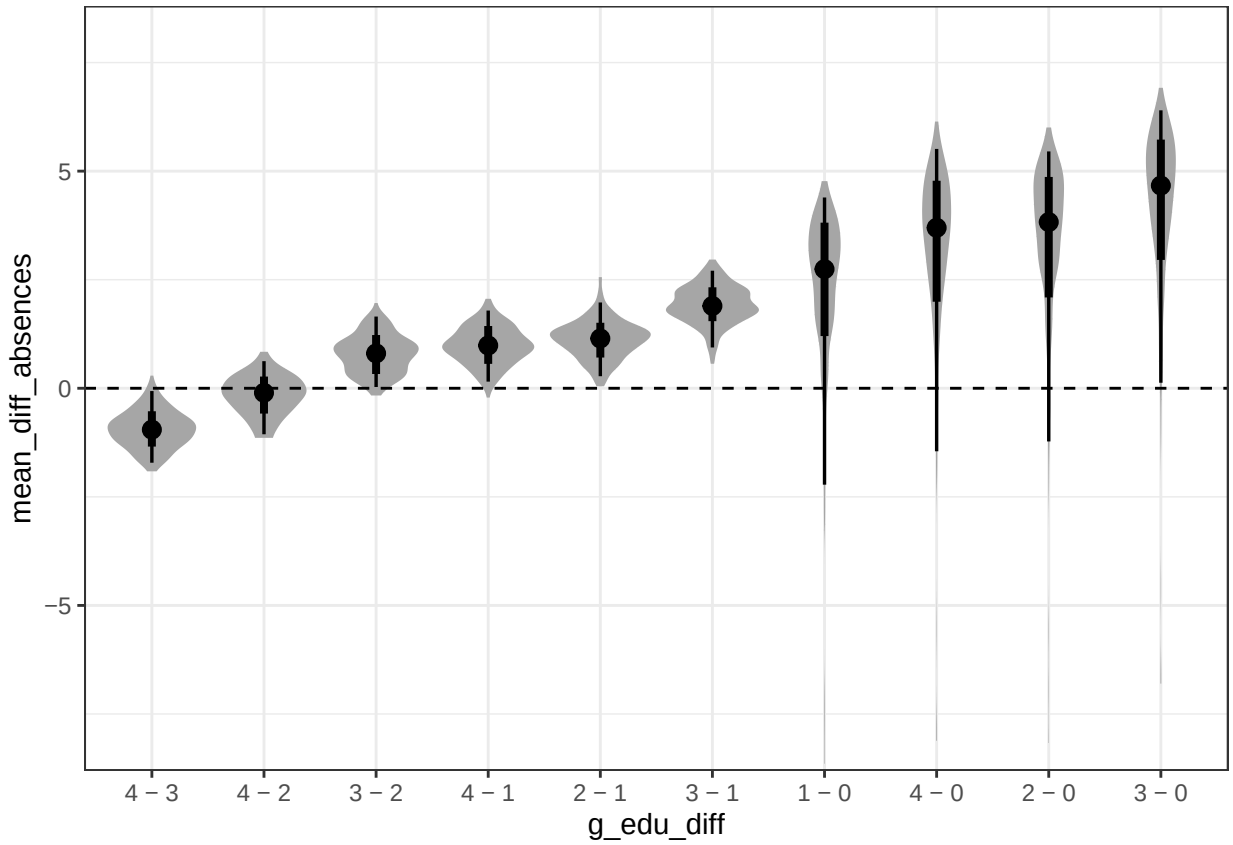
We can also use the function `compare_levels` to compute expected pairwise differences in absences between levels of guardian education. These contrasts are akin to a t-test for the difference between levels of a categorical variable.

```
m_main_g_edu_contrasts = predictor_grid |>
  add_epred_draws(m_mains, ndraws = 200) |>
  group_by(g_edu, .draw) |>              # marginalization
  summarise(.epred = mean(.epred)) |>    # marginalization
  compare_levels(.epred, by = g_edu) |>
  ungroup() |>
  mutate(g_edu = reorder(g_edu, .epred)) |>
  rename(
    g_edu_diff = g_edu,
    mean_diff_absences = .epred
```

```
  )
```

```
## `summarise()` has grouped output by 'g_edu'. You can override using the
## `.groups` argument.
```

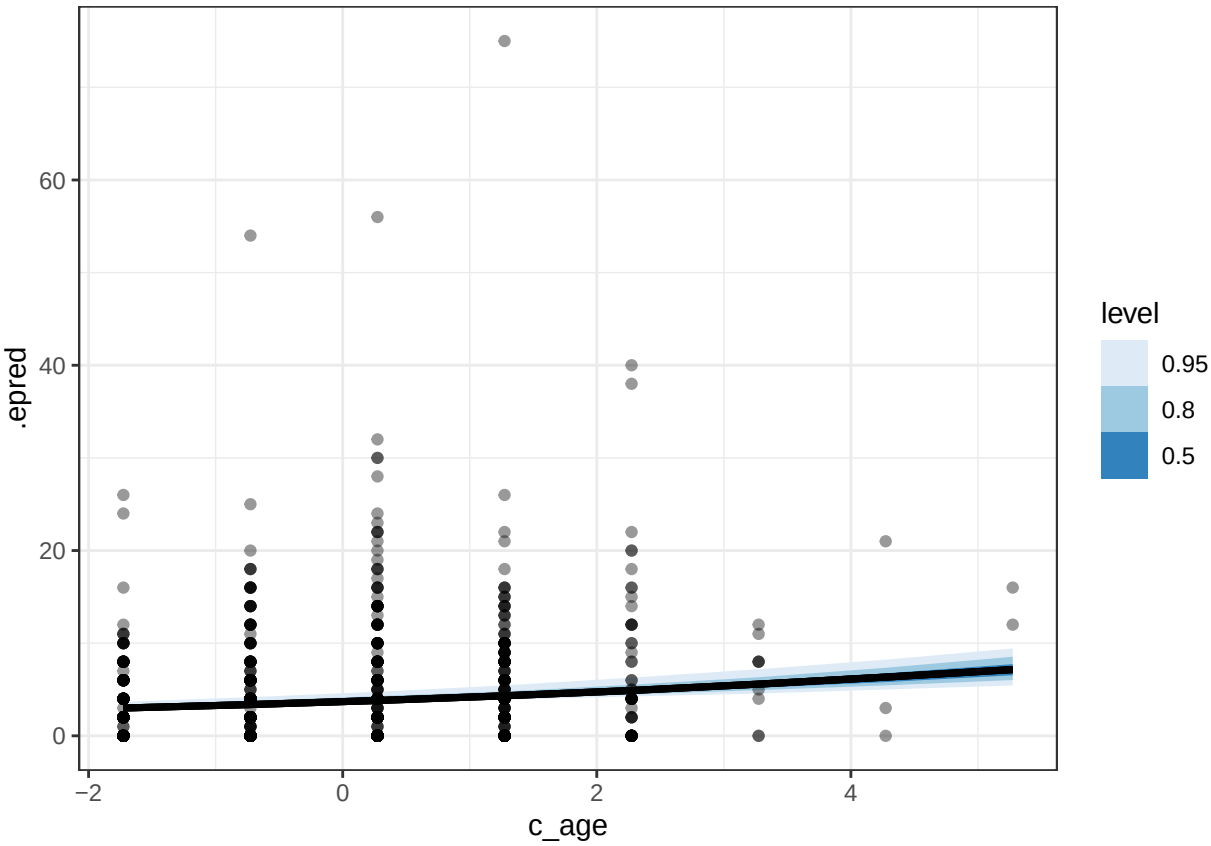
```
# showing contrasts
m_main_g_edu_contrasts |>
  ggplot(aes(x = g_edu_diff, y = mean_diff_absences)) +
  stat_eye() +
  geom_hline(yintercept = 0, linetype = "dashed") +
  coord_cartesian(ylim = c(-8, 8)) +
  theme_bw()
```



We can similarly use marginalization to summarize the impact of age on student absences. Notice how the line ribbon looks much narrower than in the predictive checks above.

```
# inferential uncertainty in age relationship
predictor_grid |>
  add_epred_draws(m_mains, ndraws = 200) |>
  group_by(c_age, .draw) |>              # marginalization
  summarise(.epred = mean(.epred)) |>    # marginalization
  ggplot(aes(x = c_age, y = .epred)) +
  stat_lineribbon(.width = c(.50, .80, .95)) +
  scale_fill_brewer() +
  geom_point(aes(y = absences), alpha = 0.4, data = model_df) +
  theme_bw()
```

```
## `summarise()` has grouped output by 'c_age'. You can override using the
## `.groups` argument.
```


60
40
20
0
.epred
−2
0
2
4
c_age
level
0.95
0.8
0.5