Week 10 Exercises

Research Paper. Continue to make progress on your research paper. The next draft is due Tuesday of Thanksgiving week. This draft should be close to finalized, so that I can make suggestions to help polish the paper.

Workshop. I want to distribute some more comments on the workshop materials. I've got a few ideas. I'll need a little time, so hold off on rehearsing for now.

Exercise 1 Simulate and recover; negative binomial

One important way to test and develop your understandings of statistical models and quantities of interest is to *simulate* a fake data set with known true quantities of interest (e.g., coefficients or first difference) and then *recover* those quantities of interest though your estimation procedure.

Do this for a negative binomial regression.

Quantities of Interest

As your quantity of interest, use the percent increase in the expected count. That's $\frac{E(Y|X_{hi})}{E(Y|X_{lo})} - 1$.

1. Compute this percent increase setting the non-focal variables at their means (or modes, if qualitative).
2. Compute the "average percent increase" setting the non-focal variables at all observed values and then averaging the estimates.

Simulate

Simulate a fake data set with a known quantity of interest. 1. Choose a number of observations. 1. Create several predictors. 1. Choose values for all the parameters (each β and the single θ in the case of the NB). 1. Simulate an outcome variable.

Recover

1. Fit the negative binomial model and check that you have recovered the model parameters (each β and the single θ).
2. Use {marginaleffects} to recover the two quantities of interest described above.
    a. The percent increase for a typical case.
    b. The average percent increase across the observed values.

TBA.

Exercise 2 Feelings toward Donald Trump, continued

The dataset here from the 2016 ANES has data on feeling thermometer ratings of Donald Trump and a `social_group` variable that indicates a race-sex-degree triplet.

```
anes <- read_csv("data/anes-ft-groups.csv")
```

I'm interested in the feelings of each group toward Donald Trump. You can see that the sample averages and sample sizes vary a lot across the groups. In a previous homework, we used a hierarchical model to estimate the average for each group.

But rather than use a model like

```
f_simple <- ft_donald_trump ~ (1 | social_group)
```

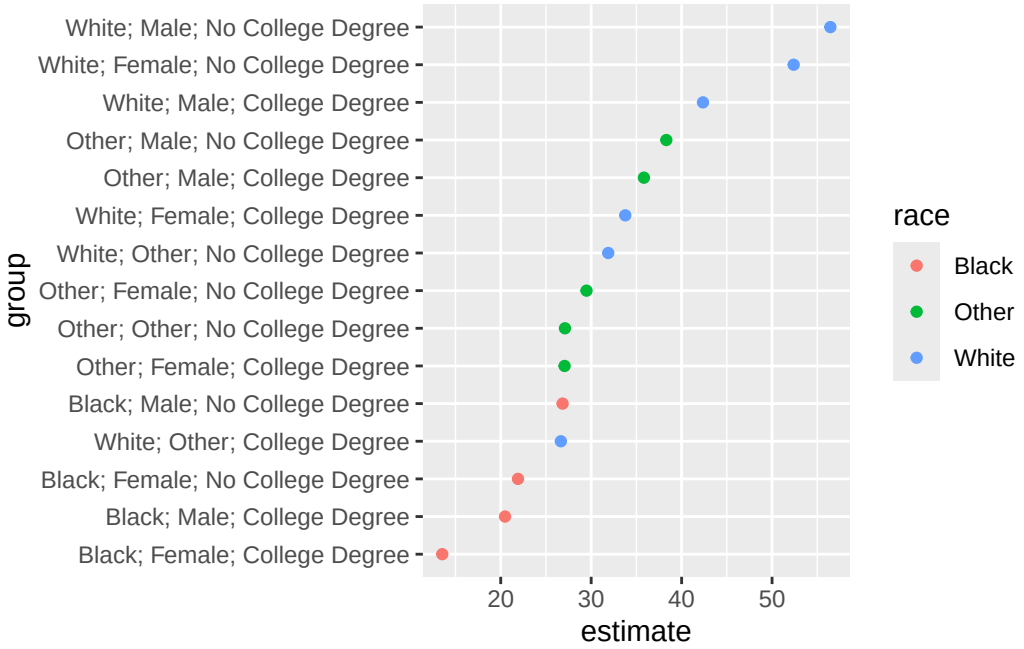
we could use a model like

```
f_complex <- ft_donald_trump ~
  (1 | race) +
  (1 | sex) +
  (1 | college_degree) +
  (1 | race:sex) +
  (1 | race:college_degree) +
  (1 | sex:college_degree) +
  (1 | race:sex:college_degree)
```

This model is rather complicated! Consider white women with a college degree. For this respondent, the model has:

1. an overall intercept
2. a deviation particular to women
3. a deviation particular to white respondents
4. a deviation particular to respondents with a college degree
5. a deviation particular to white women
6. a deviation particular to white respondents with a college degree
7. a deviation particular to women with a college degree
8. a deviation particular to white women with a college degree

Why would we do this? In some cases, certain random effects might cluster close together. The figure below shows that black respondents tend to have cooler feelings toward Donald Trump (for all values of sex and education).



In this case, it might not be a great choice to pool the four groups of black respondents toward an overall mean. Instead, we should pool them toward each other. The model below captures that logic explicitly with respect to race only.

```
f_slightly_complex <- ft_donald_trump ~
  (1 | race) +
  (1 | race:sex:college_degree)
```

The `f_complex` model above replicates this logic for all the variables, not race only.

As you're exploring what clusters might be helpful, you can compute the LOOIC for modeled fitted with `brm()` easily. See `?brms::loo_compare` for an example. *IC and Stan are evolving rapidly; see `?loo::`loo-package` for the latest details.

The BIC rule of thumb of >10 I recommended earlier doesn't apply for LOOIC. When comparing models with LOOIC, interpret the difference in expected log predictive density relative to its standard error. If the difference is greater than two SE, the difference is clearly meaningful.

Solution.

```
fit_simple <- brm(f_simple,
                  data = anes,
                  chains = 10,
                  cores = 10,
                  iter = 2000,
                  warmup = 1000) |>
  add_criterion("loo")
```

```
fit_complex <- brm(f_complex,
                   data = anes,
                   chains = 10,
                   cores = 10,
                   iter = 4000,
                   warmup = 2000) |>
  add_criterion("loo")
```

```
fit_slightly_complex <- brm(f_slightly_complex,
                            data = anes,
                            chains = 10,
                            cores = 10,
                            iter = 2000,
                            warmup = 1000) |>
  add_criterion("loo")
```

```
f4 <- ft_donald_trump ~
  (1 | race) +
  (1 | sex) +
  (1 | race:sex) +
  (1 | college_degree) +
  (1 | college_degree:race)
fit4 <- brm(f4,
            data = anes,
```

```
            chains = 10,
            cores = 10,
            iter = 2000,
            warmup = 1000) |>
  add_criterion("loo")
```

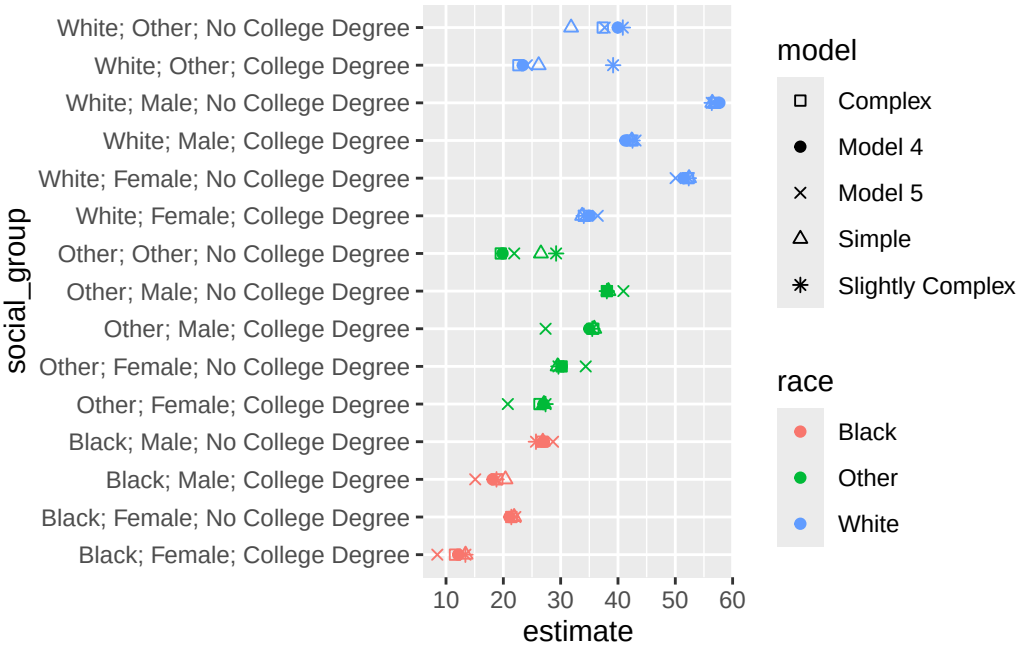
```
f5 <- ft_donald_trump ~
  (1 | race) +
  (1 | sex) +
  (1 | college_degree)
fit5 <- brm(f5,
            data = anes,
            chains = 10,
            cores = 10,
            iter = 2000,
            warmup = 1000) |>
  add_criterion("loo")
```

```
loo_compare(
  fit_simple,
  fit_complex,
  fit_slightly_complex,
  fit4,
  fit5,
  criterion = "loo")
```

```
                    elpd_diff se_diff
fit4                 0.0       0.0
fit_complex         -0.3       1.5
fit_simple          -1.7       2.0
fit_slightly_complex -1.9       1.9
fit5                -9.0       4.4
```

```
grid <- anes |>
  select(race, sex, college_degree, social_group) |>
  distinct()

p1 <- predictions(fit_simple, newdata = grid) |>
  mutate(model = "Simple")
p2 <- predictions(fit_complex, newdata = grid) |>
  mutate(model = "Complex")
```

```r
p3 <- predictions(fit_slightly_complex, newdata = grid) |>
  mutate(model = "Slightly Complex")
p4 <- predictions(fit4, newdata = grid) |>
  mutate(model = "Model 4")
p5 <- predictions(fit5, newdata = grid) |>
  mutate(model = "Model 5")
p <- bind_rows(p1, p2, p3, p4, p5)
ggplot(p, aes(x = estimate, y = social_group, color = race, shape = model)) +
  geom_point() +
  scale_shape_manual(values = c(0, 19, 4, 2, 8))
```



Exercise 3 Tappin (2023)

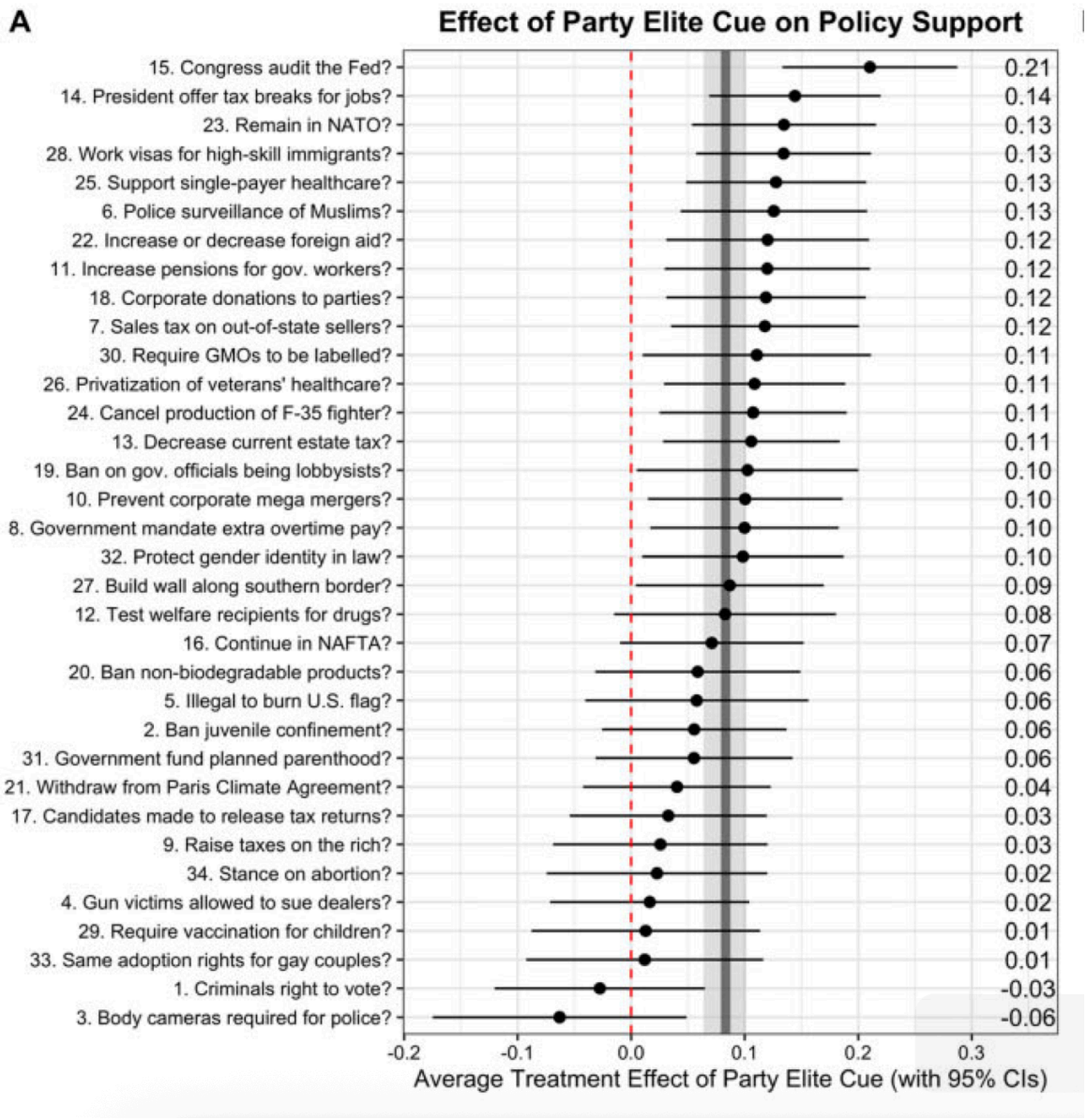
Background

"Elite cues" occur when citizens learn where partisan leaders stand on a policy. These cues sometimes have large effects on policy attitudes. Tappin (2023) examines why the estimated effects of party elite cues vary so widely across studies. He conducts a survey experiment using 1,700 Americans. He randomly assigns respondents to receive a cue or not, but also *randomly assigns respondents to policies*. He shows that treatment effects differ substantially

by policy. This highlights the importance of modeling across-policy heterogeneity in treatment effects. Scott and I argue for similar designs in Clifford, Leeper, and Rainey (2024); Clifford and Rainey (2024); and Clifford and Rainey (2025).

Tasks

1. Skim Tappin (2023) to understand his Figure 2A.
2. Reproduce Tappin's (2023) Figure 2A using least squares without pooling across policies. You don't need to include all details or reproduce the same formatting; just similarly plot the estimated treatment effects and CIs.
3. Estimate the same treatment effects using the hierarchical model. Explain the differences between the figures. Do you think we should prefer the hierarchical model?



Details

Data are available in this Dropbox folder.

- `outcome_recode_01` is the outcome variable
- `cue` is the treatment indicator
- `item` indexes the policy
- `pid` indexes respondents.

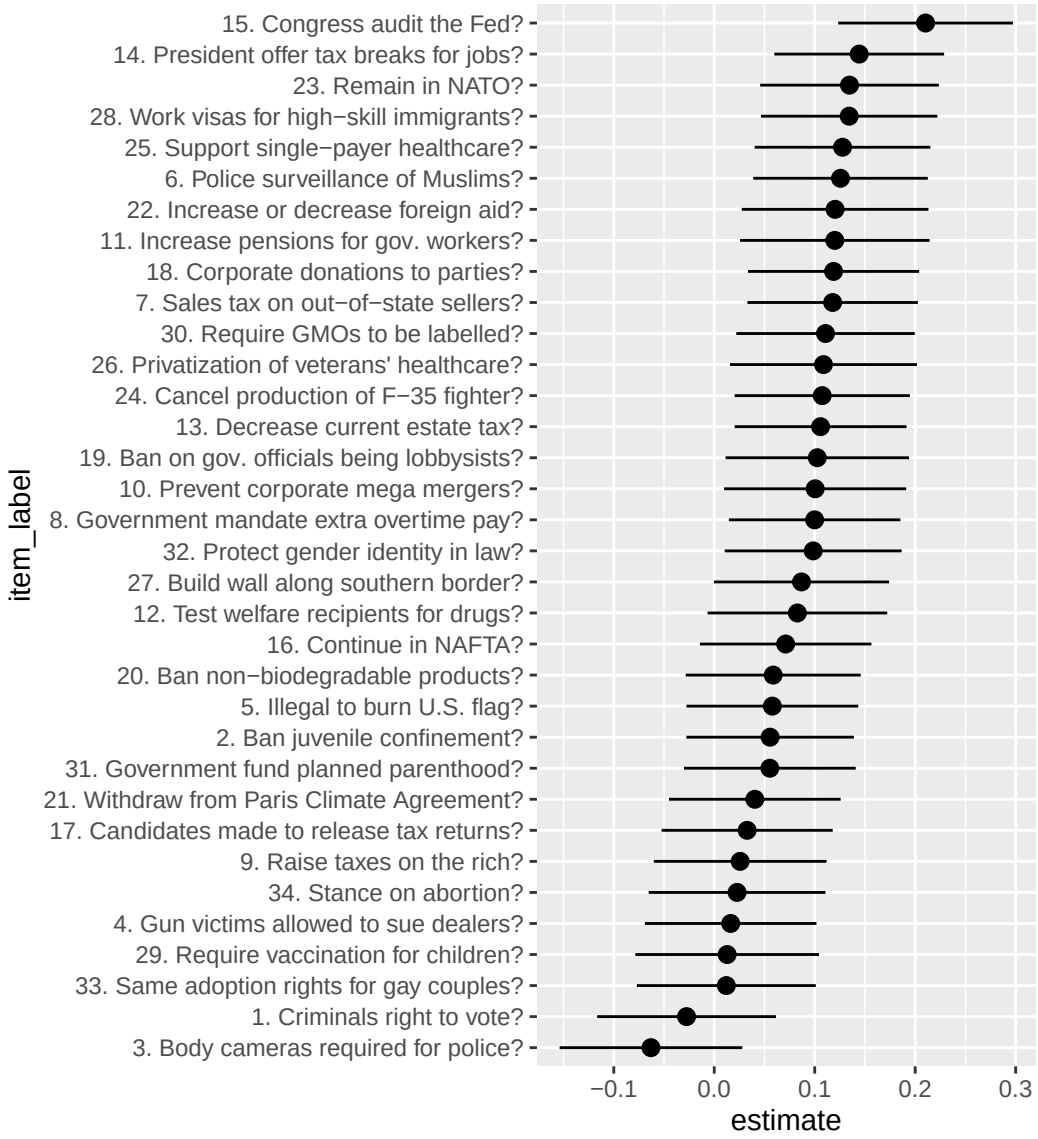
You want to fit the hierarchical model `outcome_recode_01 ~ 1 + cue + (1 + cue | item_label)`.

```
# load Tappin's (2023) data
# - from https://osf.io/t2bpj/
df <- read_rds("data/open_data.rds")

# following Tappin, do a little wrangling
# - from analysis__original_exp.R at https://osf.io/t2bpj/
df_for_model <-
  df %>%
  filter(continue_check == 2,
         item %in% c(1:34)) %>%
  filter(pol_party != 4) %>%
  select(pid,
         outcome_recode_01,
         cue,
         item,
         item_label,
         word_captcha,
         policy_group,
         order_variable) %>%
  filter(!is.na(outcome_recode_01)) |>
  glimpse()
```

```
Rows: 7,460
Columns: 8
$ pid               <int> 5, 5, 5, 5, 5, 7, 7, 7, 7, 7, 7, 7, 8, 8, 8, 8, 8, 8~
$ outcome_recode_01 <dbl> 0.8333333, 0.6666667, 0.0000000, 0.6666667, 0.166666~
$ cue               <int> 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1~
$ item              <dbl> 1, 4, 6, 9, 18, 1, 4, 5, 13, 14, 30, 34, 4, 5, 8, 17~
$ item_label        <chr> "1. Criminals right to vote?", "4. Gun victims allow~
$ word_captcha      <chr> "hello", "hello", "hello", "hello", "hello", "hello"~
$ policy_group      <chr> "Crime", "Domestic policy", "Domestic policy", "Econ~
$ order_variable    <dbl> 4, 1, 6, 8, 5, 2, 4, 6, 3, 7, 5, 1, 4, 6, 8, 2, 7, 3~
```

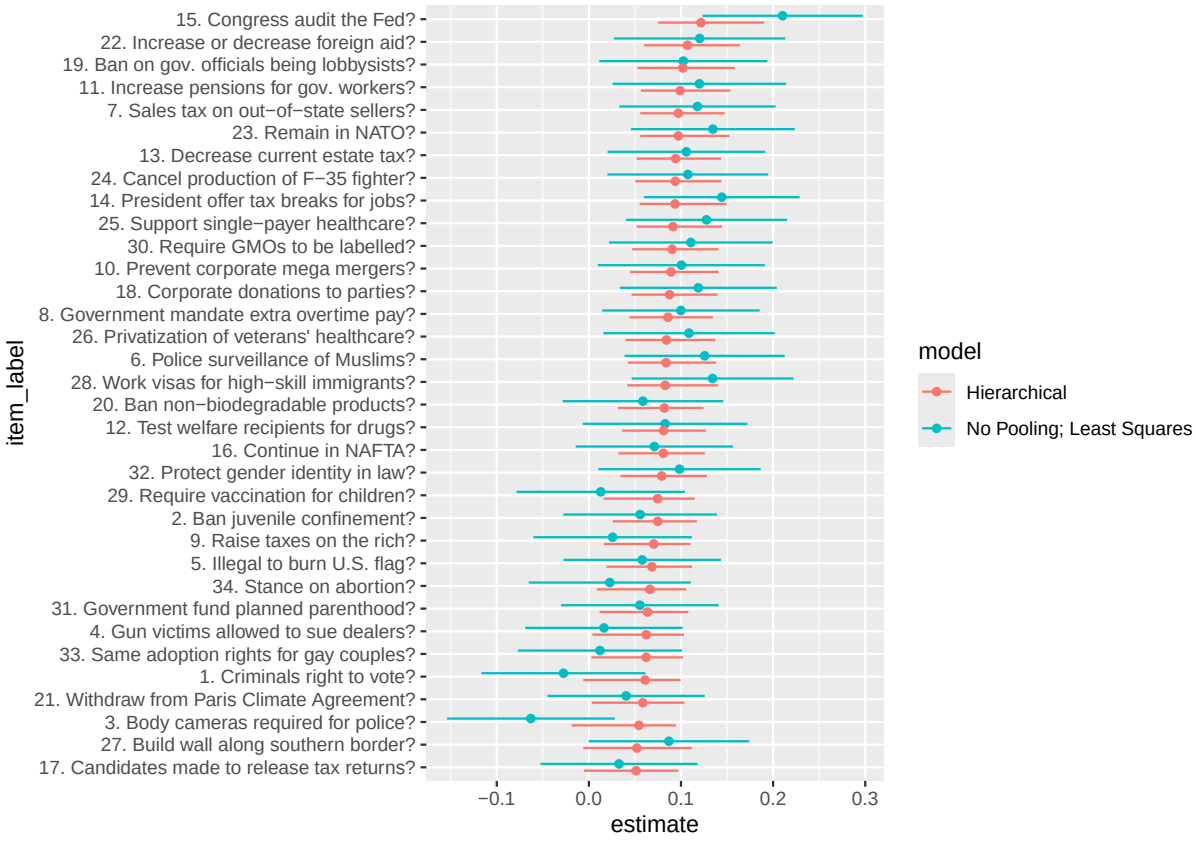
Solution.

```r
ls_fit <- lm(outcome_recode_01 ~ item_label*cue, data = df_for_model)

c <- comparisons(ls_fit,
                 variables = list(cue = c(0, 1)),
                 newdata = datagrid(item_label = unique)) |>
  mutate(item_label = reorder(item_label, estimate)) |>
  mutate(model = "No Pooling; Least Squares")

ggplot(c, aes(x = estimate, y = item_label, xmin = conf.low, xmax = conf.high)) +
  geom_pointrange()
```



```r
brm_fit <- brm(outcome_recode_01 ~ 1 + cue + (1 + cue | item_label),
               data = df_for_model,
               chains = 10,
               cores = 10,
               iter = 2000,
               warmup = 1000)
```

```r
c2 <- comparisons(brm_fit,
                  variables = list(cue = c(0, 1)),
                  newdata = datagrid(item_label = unique)) |>
  mutate(item_label = reorder(item_label, estimate)) |>
  mutate(model = "Hierarchical") |>
  bind_rows(c)

ggplot(c2, aes(x = estimate,
               y = item_label,
               xmin = conf.low,
               xmax = conf.high,
               color = model)) +
  geom_pointrange(position = ggstance::position_dodgev(height = 0.6),
                  size = 0.2)
```



Exercise 4 Simulate and recover; hiearchical linear model with varying intercepts and slops

Simulate a fake data set for the following model. Notice that theses are simple linear regression models (i.e., a slope and intercept) for each group.

$$y_{ij} \sim N(\mu_{ij}, \sigma_y^2) \tag{1}$$

$$\mu_{ij} = \beta_j^{\text{cons}} + \beta_j^x x_i \tag{2}$$

$$\begin{pmatrix} \beta_j^{\text{cons}} \\ \beta_j^x \end{pmatrix} \sim MVN\left(\begin{pmatrix} \mu_{\beta^{\text{cons}}} \\ \mu_{\beta^x} \end{pmatrix}, \begin{pmatrix} \sigma^2_{\beta^{\text{cons}}} & \rho\sigma_{\beta^{\text{cons}}}\sigma_{\beta^x} \\ \rho\sigma_{\beta^{\text{cons}}}\sigma_{\beta^x} & \sigma^2_{\beta^x} \end{pmatrix} \right) \tag{3}$$

In this case, j indexes the group of respondents and i indexes the respondents. There is a respondent-level explanatory variable x_i. Use about 100 respondents per group and about 100 groups. Show that you can use `lmer()` to recover σ_y^2, $\mu_{\beta^{\text{cons}}}$, μ_{β^x}, ρ, $\sigma_{\beta^{\text{cons}}}$, and σ_{β^x}. Also show that you can recover the true β_js.

Hints:

1. You need to select values for σ_y^2, $\mu_{\beta^{\text{cons}}}$, μ_{β^x}, ρ (a correlation parameter from -1 to 1), $\sigma_{\beta^{\text{cons}}}$, and σ_{β^x}.
2. *Then* simulate the β_js from `MASS:rmvnorm()`.
3. *Then* simulate the y_{ij}s (perhaps create the x_{ij}s here as well).

Solution.

```
# ---- setup

# set seed
set.seed(123)

# load packages
library(tidyverse)
library(lme4)

# ---- parameters

J <- 100   # number of groups
n_per_j <- 100   # respondents per group

sigma2_y <- 1.5 # residual variance
sigma_y <- sqrt(sigma2_y)

mu_beta_cons <- 0.5  # mean intercept
mu_beta_x <- 1.2  # mean slope
sd_beta_cons <- 0.8  # SD of intercepts
sd_beta_x <- 0.6  # SD of slopes
rho_beta <- 0.35 # corr(intercept, slope) in the (beta_cons, beta_x) vector

# ---- building the needed matrices and vectors from above

Sigma_beta <- matrix(
  c(sd_beta_cons^2, rho_beta * sd_beta_cons * sd_beta_x,
    rho_beta * sd_beta_cons * sd_beta_x, sd_beta_x^2),
  nrow = 2, byrow = TRUE
)
```

```r
mu <- c(mu_beta_cons, mu_beta_x)

# draw the group-level (true) coefficients once
beta <- MASS::mvrnorm(n = J, mu = mu, Sigma = Sigma_beta)

# ---- simulate by group using a for-loop

# simulate data for each group
data_list <- list()
for (j in 1:J) {
  # pull this group's true parameters
  b0_j <- beta[j, 1]
  b1_j <- beta[j, 2]

  # respondent-level x (can be whatever you like; here: standard normal)
  x_ij <- rnorm(n_per_j, mean = 0, sd = 1)

  # linear predictor and outcome
  mu_ij <- b0_j + b1_j * x_ij
  y_ij  <- rnorm(n_per_j, mean = mu_ij, sd = sigma_y)

  data_list[[j]] <- tibble(
    group = j,
    i = 1:n_per_j,
    x = x_ij,
    y = y_ij
  )
}

# bind_rows after the loop
data <- bind_rows(data_list)
```

```r
# ---- recover

# fit model
fit <- lmer(y ~ 1 + x + (1 + x | group), data = data)

# summary
summary(fit)
```

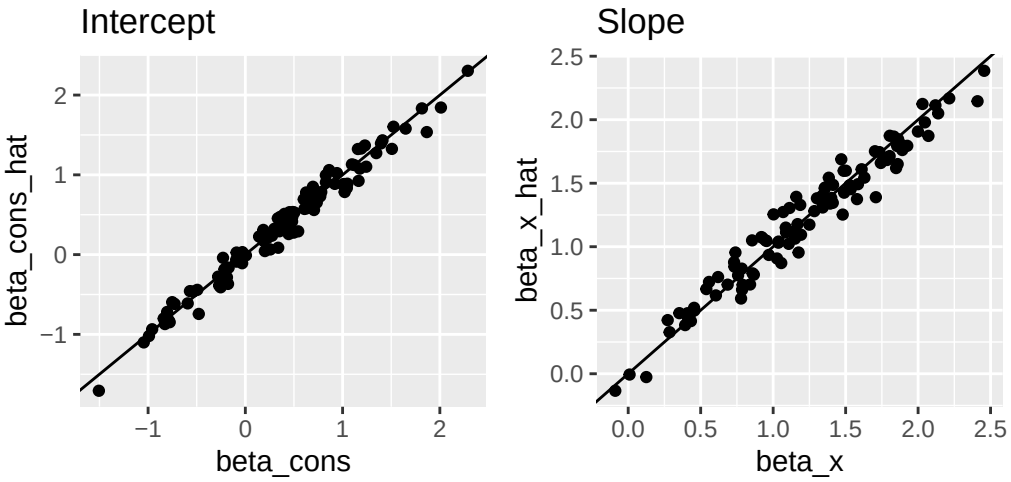
```
Linear mixed model fit by REML ['lmerMod']
Formula: y ~ 1 + x + (1 + x | group)
```

```
   Data: data

REML criterion at convergence: 33168.3

Scaled residuals:
    Min      1Q  Median      3Q     Max
-3.6143 -0.6636  0.0077  0.6654  3.7973

Random effects:
 Groups   Name        Variance Std.Dev. Corr
 group    (Intercept) 0.5638   0.7509
          x           0.2905   0.5390   0.27
 Residual             1.5110   1.2292
Number of obs: 10000, groups:  group, 100

Fixed effects:
            Estimate Std. Error t value
(Intercept)  0.38603    0.07609   5.073
x            1.21323    0.05532  21.933

Correlation of Fixed Effects:
  (Intr)
x 0.260
```

```
# Fixed effects:
# (Intercept) = 0.386  → _{ ^cons}  (true 0.5)
# x            = 1.213  → _{ ^x}     (true 1.2)

# Random effects (group):
# Var(Intercept) = 0.5638 →  ²_{ ^cons} (true 0.64)
# Var(x)         = 0.2905 →  ²_{ ^x}    (true 0.36)
# Corr           = 0.27   →             (true 0.35)

# Residual:
# Var(Residual)  = 1.5110 →  ²_y        (true 1.5)

# N = 100 groups × 100 obs/group = 10,000 total
```

```
fe <- fixef(fit)
mu0_hat <- fe[1]
mu1_hat <- fe[2]
```

```r
re_group <- ranef(fit)$group %>%
  as_tibble(rownames = "group") %>%
  rename(b0_dev_hat = `(Intercept)`,
         b1_dev_hat = x)

beta_hat <- re_group %>%
  mutate(
    beta_cons_hat = mu0_hat + b0_dev_hat,
    beta_x_hat = mu1_hat + b1_dev_hat
  ) %>%
  select(group, beta_cons_hat, beta_x_hat)

# compare to truth
beta_true <- data.frame(group = as.character(1:J),
                        beta_cons = beta[, 1],
                        beta_x = beta[, 2])
comp <- left_join(beta_true, beta_hat)
gg1 <- ggplot(comp, aes(x = beta_cons, y = beta_cons_hat)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1) +
  labs(title = "Intercept")
gg2 <- ggplot(comp, aes(x = beta_x, y = beta_x_hat)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1) +
  labs(title = "Slope")
library(patchwork)
gg1 + gg2
```



Exercise 5 Rich State, Poor State, Red State, Blue State

1. Read Gelman et al. (2007). *This is a great example of descriptive work and hierarchical modeling.* These authors wrote an entire book about these patterns.
2. Use the data from the 2024 ANES here to replicate (or not!) the patterns they find in Figures 4 and 5.

Variables

- `state_name`: U.S. state name joined from state_income by FIPS state_code.
- `state_median_income` State median household income (in thousands of 2024 dollars) from state_income. Example: `92.1` = $92,100.
- `rs_state_median_income`: Rescaled version of state_median_income using arm::rescale() (mean 0, SD 0.5).; i.e.,
- `income`: Numeric scores corresponding to income brackets so that (1) higher numeric values indicate higher household income and (2) the middle is zero.

ANES Category	Income Range (2024 USD)	Recode
1	Under $9,999	−2.5
2	$10,000–$29,999	−1.5
3	$30,000–$59,999	−0.5
4	$60,000–$99,999	0.5
5	$100,000–$249,999	1.5
6	$250,000 or more	2.5

- `vote_rep`: 0/1 indicator of voting for the Republican candidate Donald Trump. Derived from ANES presidential vote V242067 after filtering to major-party voters only.

```
# load data
anes <- read_csv("data/red-state.csv") |>
  glimpse()
```

```
Rows: 3,365
Columns: 5
$ state_name             <chr> "Oklahoma", "Virginia", "Colorado", "Wisconsin"~
$ state_median_income    <dbl> 66.1, 92.1, 97.1, 77.5, 77.5, 92.1, 81.1, 104.8~
$ rs_state_median_income <dbl> -0.59534222, 0.41221845, 0.60598012, -0.1535656~
```

```
$ income                <dbl> 1.5, 1.5, -0.5, -0.5, 1.5, 2.5, -0.5, 2.5, 1.5,~
$ vote_rep              <dbl> 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,~
```

Based on the regressions below, we can already see that things are different—the hierarchical model will let you dig deeper.

```
# rich individuals are less likely to vote for Trump
fit_indiv <- glm(vote_rep ~ income, data = anes, family = binomial)
arm::display(fit_indiv)
```

```
glm(formula = vote_rep ~ income, family = binomial, data = anes)
            coef.est coef.se
(Intercept) -0.21     0.04
income      -0.13     0.03
---
  n = 3365, k = 2
  residual deviance = 4582.1, null deviance = 4605.9 (difference = 23.8)
```

```
# individuals in rich states are less likely to vote for Trump
fit_state <- glm(vote_rep ~ rs_state_median_income, data = anes, family = binomial)
arm::display(fit_state)
```

```
glm(formula = vote_rep ~ rs_state_median_income, family = binomial,
    data = anes)
                       coef.est coef.se
(Intercept)            -0.25     0.04
rs_state_median_income -0.66     0.08
---
  n = 3365, k = 2
  residual deviance = 4535.4, null deviance = 4605.9 (difference = 70.4)
```

Hint: The model you want is: `vote_rep ~ rs_state_median_income*income + (1 + income | state_name)`. This model captures the following intuition: "the effect of income on vote choice is different-but-similar across states, and these differences are predicted fairly well by state income."

Solution.

```r
# load packages
library(tidyverse)
library(brms)
library(marginaleffects)

# load data
anes <- read_csv("data/red-state.csv") |>
  glimpse()
```

```
Rows: 3,365
Columns: 5
$ state_name             <chr> "Oklahoma", "Virginia", "Colorado", "Wisconsin"~
$ state_median_income    <dbl> 66.1, 92.1, 97.1, 77.5, 77.5, 92.1, 81.1, 104.8~
$ rs_state_median_income <dbl> -0.59534222, 0.41221845, 0.60598012, -0.1535656~
$ income                 <dbl> 1.5, 1.5, -0.5, -0.5, 1.5, 2.5, -0.5, 2.5, 1.5,~
$ vote_rep               <dbl> 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,~
```

```r
# fit hierarchical model
f <- vote_rep ~ rs_state_median_income*income + (1 + income | state_name)
fit <- brm(
  f,
  data = anes,
  family = bernoulli,
  chains = 4,
  cores = 4,
  iter = 4000
)
```

```r
# summary
summary(fit)
```

```
 Family: bernoulli
  Links: mu = logit
Formula: vote_rep ~ rs_state_median_income * income + (1 + income | state_name)
   Data: anes (Number of observations: 3365)
  Draws: 4 chains, each with iter = 4000; warmup = 2000; thin = 1;
         total post-warmup draws = 8000

Multilevel Hyperparameters:
~state_name (Number of levels: 51)
                     Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
```

```
sd(Intercept)             0.26      0.07     0.13     0.39 1.00     2630
sd(income)                0.07      0.04     0.01     0.16 1.00     2304
cor(Intercept,income)     0.50      0.41    -0.52     0.98 1.00     4734
                      Tail_ESS
sd(Intercept)             3608
sd(income)                2238
cor(Intercept,income)     4457

Regression Coefficients:
                            Estimate Est.Error l-95% CI u-95% CI Rhat
Intercept                      -0.21      0.06    -0.33    -0.09 1.00
rs_state_median_income         -0.55      0.12    -0.79    -0.32 1.00
income                         -0.10      0.03    -0.16    -0.04 1.00
rs_state_median_income:income  -0.15      0.07    -0.28    -0.02 1.00
                            Bulk_ESS Tail_ESS
Intercept                       3656     4933
rs_state_median_income          4566     5612
income                          7753     5710
rs_state_median_income:income   8550     5983

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).
```

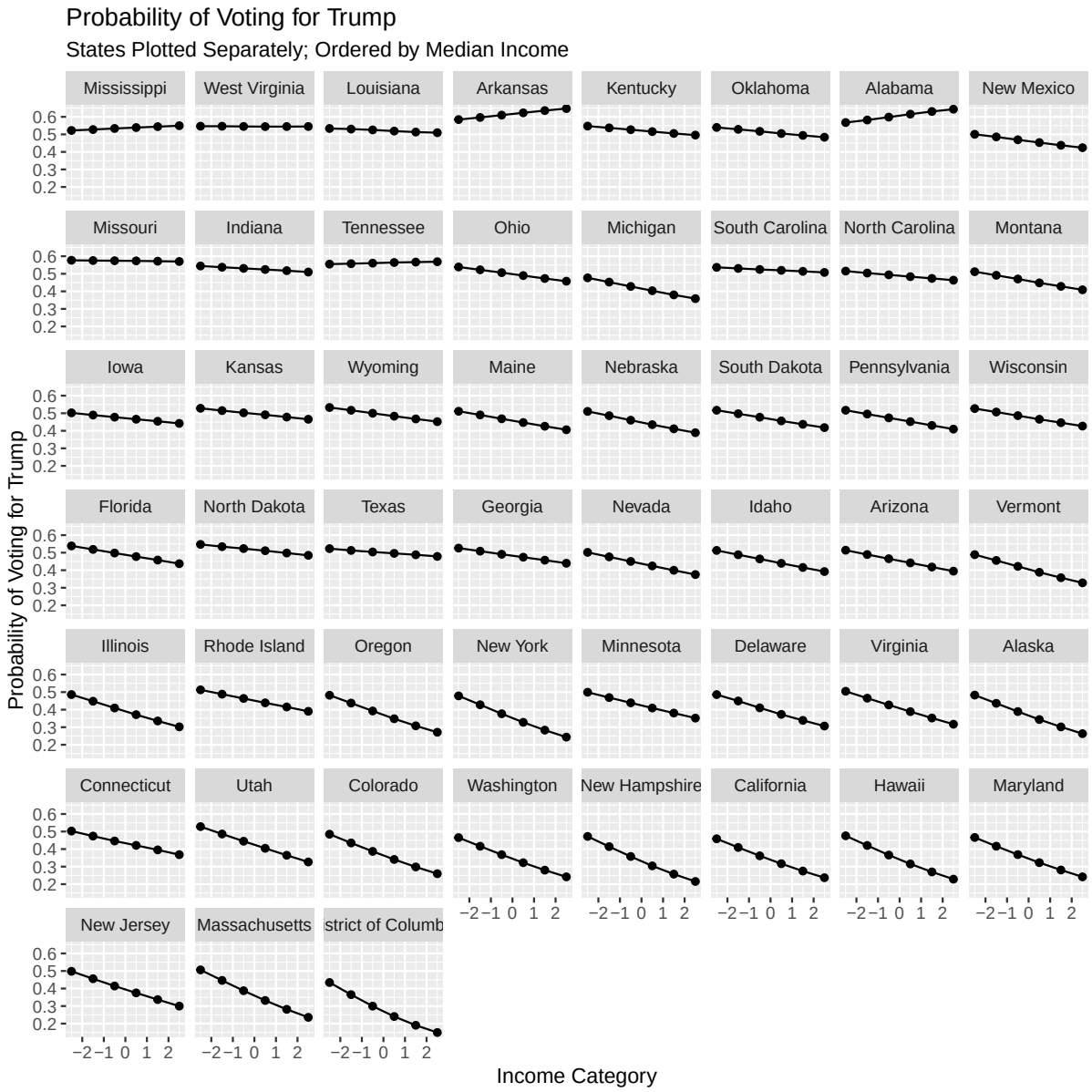
```r
# state values
states <- anes %>%
  select(state_name, state_median_income, rs_state_median_income) %>%
  distinct() %>%
  glimpse()
```

```
Rows: 51
Columns: 3
$ state_name             <chr> "Oklahoma", "Virginia", "Colorado", "Wisconsin"~
$ state_median_income    <dbl> 66.1, 92.1, 97.1, 77.5, 81.1, 104.8, 99.8, 85.8~
$ rs_state_median_income <dbl> -0.595342224, 0.412218451, 0.605980120, -0.1535~
```

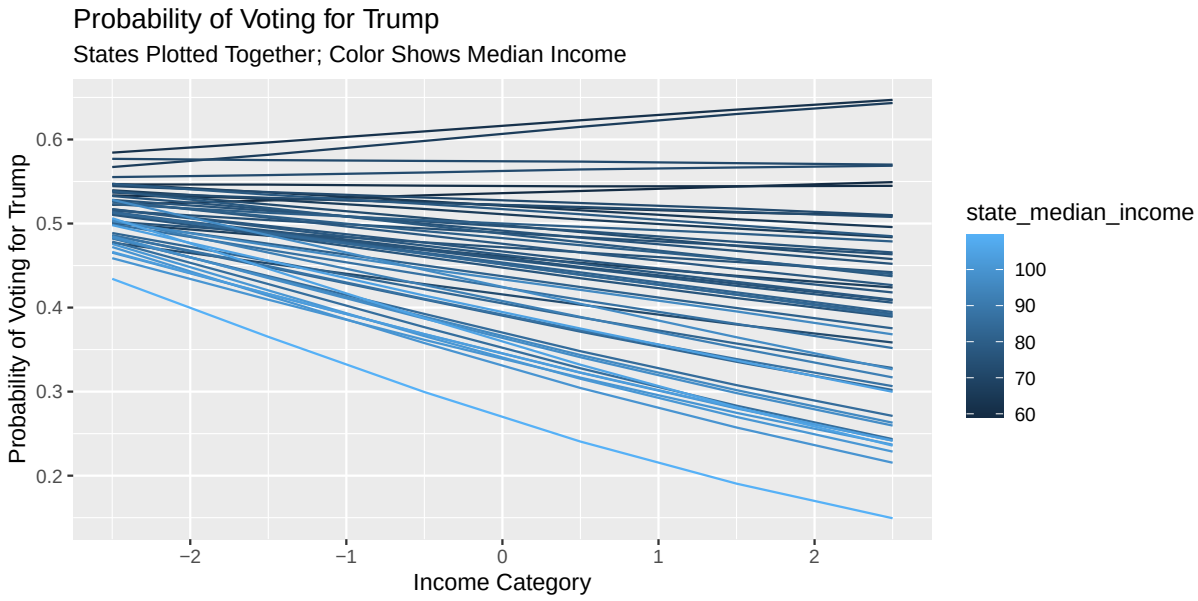
```r
# expected values
grid <- crossing(state_name = unique(anes$state_name)) |>
  left_join(states) |>
  mutate(income = NA) |>
  glimpse()
```

```
Rows: 51
Columns: 4
$ state_name             <chr> "Alabama", "Alaska", "Arizona", "Arkansas", "Ca~
$ state_median_income    <dbl> 66.7, 95.7, 81.5, 62.1, 100.1, 97.1, 96.0, 87.5~
$ rs_state_median_income <dbl> -0.572090824, 0.551726853, 0.001443714, -0.7503~
$ income                 <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
```

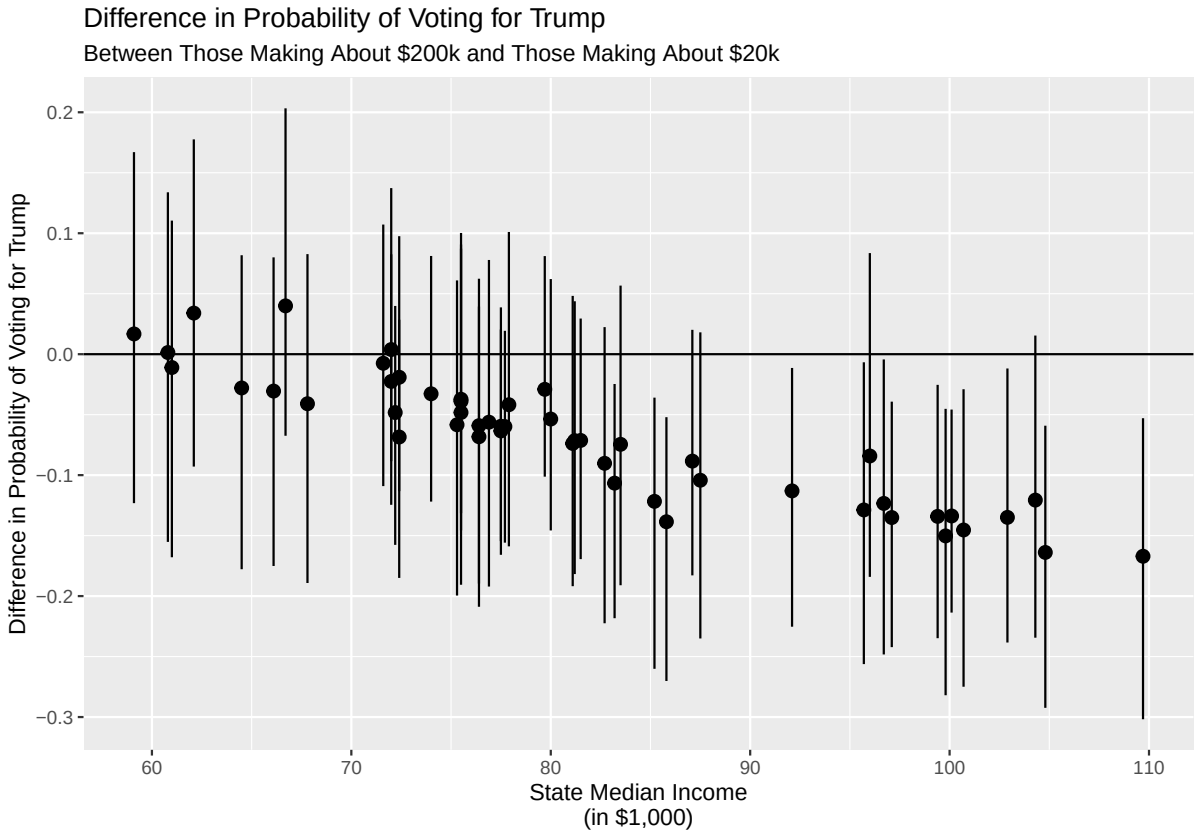
```r
p <- predictions(fit,
                 variables = list(income = unique),
                 newdata = grid) |>
  left_join(states) |>
  mutate(state_name = reorder(state_name, rs_state_median_income))
ggplot(p, aes(x = income, y = estimate, group = state_name)) +
  geom_line() +
  geom_point() +
  facet_wrap(vars(state_name)) +
  labs(x = "Income Category",
       y = "Probability of Voting for Trump",
       title = "Probability of Voting for Trump",
       subtitle = "States Plotted Separately; Ordered by Median Income")
```




```
# first differences
ggplot(p, aes(x = income, y = estimate, group = state_name, color = state_median_income)) +
  geom_line() +
  labs(x = "Income Category",
       y = "Probability of Voting for Trump",
       title = "Probability of Voting for Trump",
       subtitle = "States Plotted Together; Color Shows Median Income")
```




```
# first differences
c <- comparisons(fit,
                 variables = list(income = c(-1.5, 1.5)),
                 newdata = grid)
ggplot(c, aes(x = state_median_income,
              y = estimate,
              ymin = conf.low,
              ymax = conf.high)) +
  geom_hline(yintercept = 0) +
  geom_pointrange() +
  labs(x = "State Median Income\n(in $1,000)",
       y = "Difference in Probability of Voting for Trump",
       title = "Difference in Probability of Voting for Trump",
       subtitle = "Between Those Making About $200k and Those Making About $20k")
```


Difference in Probability of Voting for Trump
Between Those Making About $200k and Those Making About $20k
Difference in Probability of Voting for Trump
0.2
0.1
0.0
−0.1
−0.2
−0.3
60
70
80
90
100
110
State Median Income
(in $1,000)