Bootstrapping + Regression, pt. 1

Rui Chen

Overview

Today:

1. Bootstrapping (uncertainty around mean and β)
2. Basic regression modeling in R (fitting, interpreting, plotting, and conditional relationships)

The Bootstrap

Task: how often Americans eat ice cream in a given month.

Sub-task: check distributional assumptions of the *likely* distribution of these data: Poisson distribution

The probability mass function (PMF) for the Poisson distribution,

$$\Pr(X = x) = e^{-\lambda}\frac{\lambda^k}{k!},$$

where λ is the event rate, e is Euler's number, k is an integer with range $[0, \infty]$.

Bootstrapping (from lecture):

1. Draw B samples **with replacement** from the sample (of size N)
2. (For our task) calculate the mean of the bootstrapped sample means $\hat{\mu}_1, \hat{\mu}_2, \ldots, \hat{\mu}_B$
3. Estimate the standard error (SE) of the sample mean $\hat{\mu}$, $SE_B(\hat{\mu}) = \sqrt{\frac{1}{B-1}\sum_{r=1}^{B}\left(\hat{\mu}_r - \frac{1}{B}\sum_{r'=1}^{B}\hat{\mu}_{r'}\right)^2}$

Application

Let's see this in action.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 0.1.2 --
```

```
## v broom     0.7.3      v recipes   0.1.15
## v dials     0.0.9      v rsample   0.0.8
## v infer     0.5.4      v tune      0.1.2
## v modeldata 0.1.0      v workflows 0.2.1
## v parsnip   0.1.4      v yardstick 0.0.7
```

```
## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
```

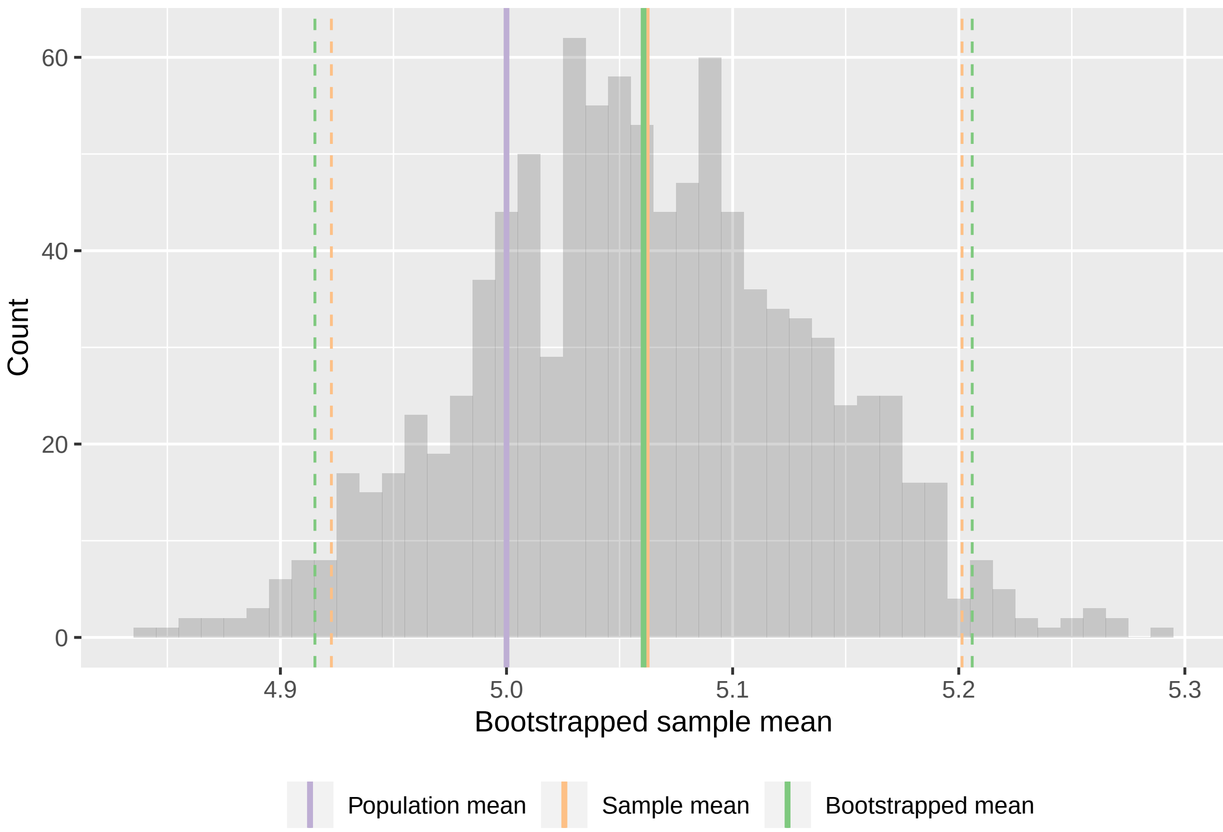
```
# set up data
set.seed(1234)

mu <- 5
n_obs <- 1000
ice <- tibble(sim = rpois(n_obs, lambda = mu))

mu_samp <- mean(ice$sim)
sem <- sqrt(mu_samp / n_obs)

# Bootstrap

## helper fun
mean_ice <- function(splits) {
  x <- analysis(splits)
  mean(x$sim)
}

ice_boot <- ice %>%
  bootstraps(1000) %>%
  mutate(mean = map_dbl(splits, mean_ice))

boot_sem <- sd(ice_boot$mean)

# compare
tibble(sem, boot_sem)
```

```
## # A tibble: 1 x 2
##      sem boot_sem
##    <dbl>    <dbl>
## 1 0.0711   0.0742
```

Now, plot.

```
ggplot(ice_boot, aes(mean)) +
  geom_histogram(binwidth = .01, alpha = 0.25) +
  geom_vline(aes(xintercept = mu, color = "Population mean"), size = 1) +
  geom_vline(aes(xintercept = mu_samp, color = "Sample mean"), size = 1) +
  geom_vline(aes(xintercept = mean(mean),
                 color = "Bootstrapped mean"), size = 1) +
  geom_vline(aes(xintercept = mean(mean) + 1.96 * boot_sem,
                 color = "Bootstrapped mean"), linetype = 2) +
```

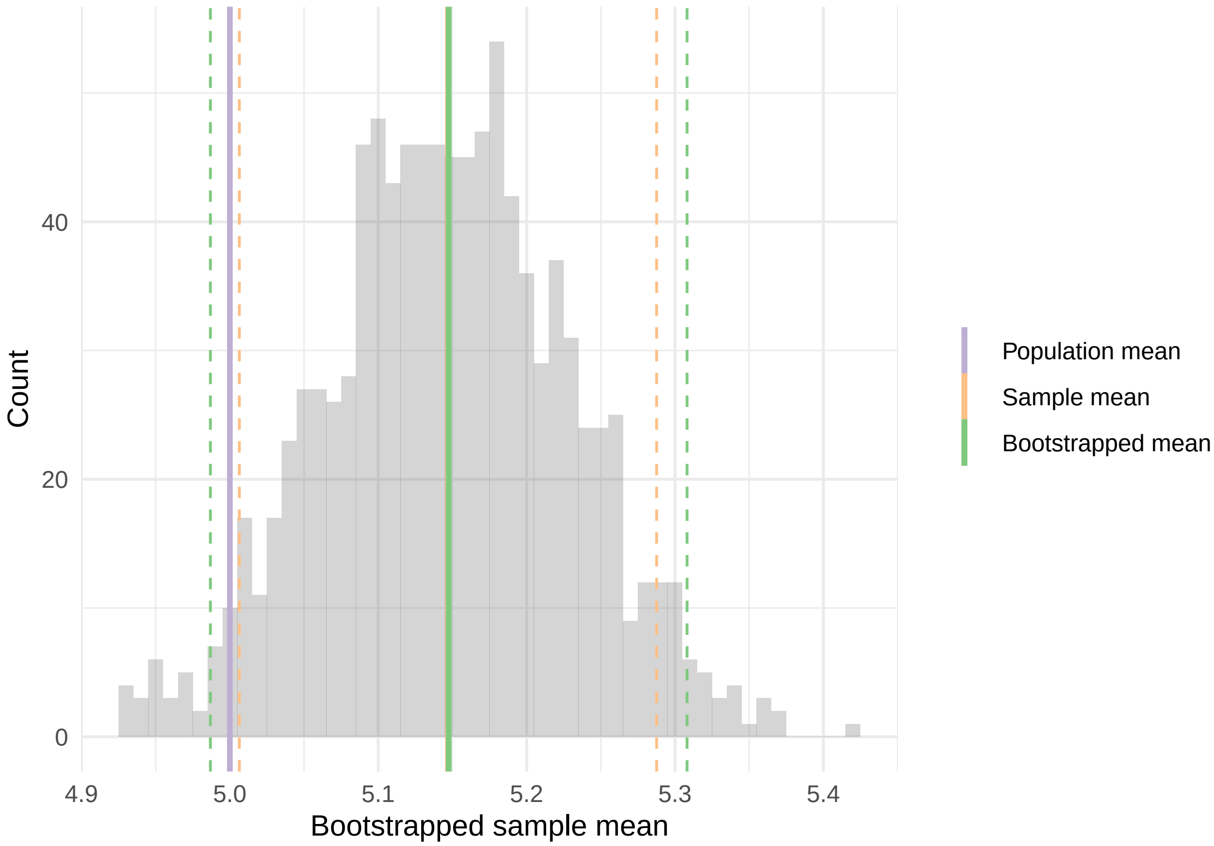
```r
  geom_vline(aes(xintercept = mean(mean) - 1.96 * boot_sem,
                 color = "Bootstrapped mean"), linetype = 2) +
  geom_vline(aes(xintercept = mu_samp + 1.96 * sem, color = "Sample mean"),
             linetype = 2) +
  geom_vline(aes(xintercept = mu_samp - 1.96 * sem, color = "Sample mean"),
             linetype = 2) +
  scale_color_brewer(type = "qual",
                     name = NULL,
                     breaks = c("Population mean", "Sample mean",
                                "Bootstrapped mean")) +
  labs(x = "Bootstrapped sample mean",
       y = "Count") +
  theme(legend.position = "bottom")
```



Now, let's break the process and violate the Poisson assumptions.

```r
# break it
set.seed(113)

ice2 <- tibble(sim = c(rpois(n_obs / 2, lambda = mu),
                       round(runif(n_obs / 2, min = 0, max = 10))))

# new calcs
mu2_samp <- mean(ice2$sim)
sem2 <- sqrt(mu2_samp / n_obs)

# bootstrapping
ice2_boot <- ice2 %>%
```

```r
  bootstraps(1000) %>%
  mutate(mean = map_dbl(splits, mean_ice))

boot2_sem <- sd(ice2_boot$mean)

# plot
ggplot(ice2_boot, aes(mean)) +
  geom_histogram(binwidth = .01, alpha = 0.25) +
  geom_vline(aes(xintercept = mu, color = "Population mean"), size = 1) +
  geom_vline(aes(xintercept = mu2_samp, color = "Sample mean"), size = 1) +
  geom_vline(aes(xintercept = mean(mean),
               color = "Bootstrapped mean"), size = 1) +
  geom_vline(aes(xintercept = mean(mean) + 1.96 * boot2_sem,
               color = "Bootstrapped mean"), linetype = 2) +
  geom_vline(aes(xintercept = mean(mean) - 1.96 * boot2_sem,
               color = "Bootstrapped mean"), linetype = 2) +
  geom_vline(aes(xintercept = mu2_samp + 1.96 * sem2, color = "Sample mean"),
             linetype = 2) +
  geom_vline(aes(xintercept = mu2_samp - 1.96 * sem2, color = "Sample mean"),
             linetype = 2) +
  scale_color_brewer(type = "qual",
                     name = NULL,
                     breaks = c("Population mean", "Sample mean",
                                "Bootstrapped mean")) +
  labs(x = "Bootstrapped sample mean",
       y = "Count") +
  theme_minimal()
```



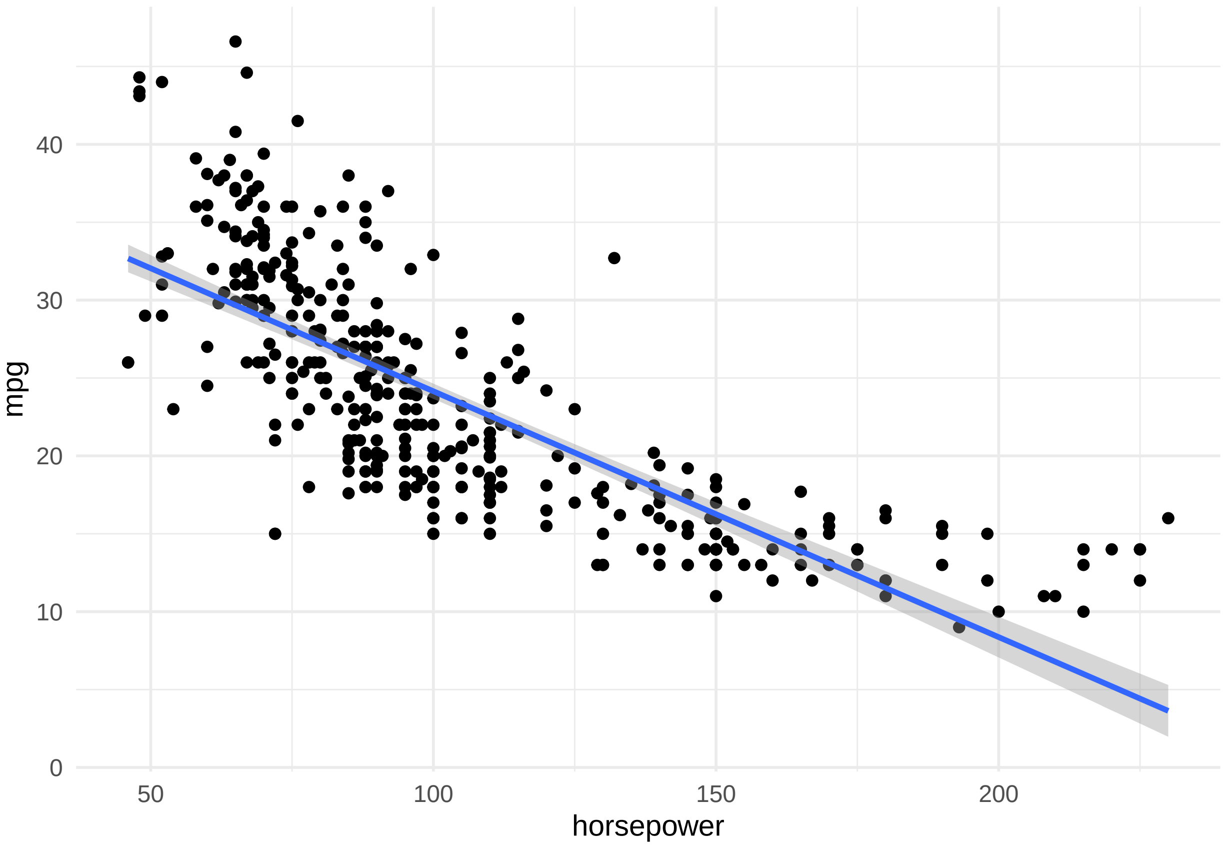
Estimating the accuracy of a linear regression model

Task: calculate uncertainty around coefficient estimates from linear regression

Back to the `horsepower` and `mpg` linear model via the `Auto` dataset.

```
library(ISLR)

Auto <- as_tibble(Auto)

# descriptive plot
ggplot(Auto, aes(horsepower, mpg)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_minimal()
```

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




```r
# SLM
auto_lm <- lm(mpg ~ poly(horsepower, 1, raw = TRUE), data = Auto); tidy(auto_lm)
```

```
## # A tibble: 2 x 5
##   term                            estimate std.error statistic   p.value
##   <chr>                              <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)                       39.9     0.717        55.7 1.22e-187
## 2 poly(horsepower, 1, raw = TRUE)   -0.158   0.00645     -24.5 7.03e- 81
```

```r
# Bootstrap
lm_coefs <- function(splits, ...) {
  mod <- lm(..., data = analysis(splits))
  tidy(mod)
}

auto_boot <- Auto %>%
  bootstraps(1000) %>%
  mutate(coef = map(splits, lm_coefs, as.formula(mpg ~ poly(horsepower, 1, raw = TRUE))))

# calc and compare
auto_boot %>%
  unnest(coef) %>%
  group_by(term) %>%
  summarize(.estimate = mean(estimate),
            .se = sd(estimate, na.rm = TRUE))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 2 x 3
```

```
##   term                          .estimate     .se
##   <chr>                             <dbl>   <dbl>
## 1 (Intercept)                      40.0   0.855
## 2 poly(horsepower, 1, raw = TRUE)    -0.158 0.00740
```

Regression & INXN

Explore basic linear models in R and conditional relationships via 2008 NES data.

```r
library(tidyverse)
library(foreign)
library(skimr)
library(broom)
library(modelr)
```

```
##
## Attaching package: 'modelr'

## The following objects are masked from 'package:yardstick':
##
##     mae, mape, rmse

## The following object is masked from 'package:broom':
##
##     bootstrap
```

```r
library(here)
```

```
## here() starts at /Users/raychanan/Github/Data-and-Code
```

```r
set.seed(1234)
theme_set(theme_minimal())

# get nes data
nes <- read.dta(here("data", "nes2008.dta")) %>%
  select(obama_therm_post, partyid3, libcon7, libcon7_obama) %>%
  mutate_all(funs(ifelse(is.nan(.), NA, .))) %>%
  rename(ObamaTherm = obama_therm_post,
         RConserv = libcon7,
         ObamaConserv = libcon7_obama) %>%
  mutate(GOP = ifelse(partyid3 == 3, 1, 0)) %>%
  select(-partyid3) %>%
  na.omit()
```

```
## Warning: `funs()` is deprecated as of dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```r
# inspect
skim(nes)
```

Table 1: Data summary

Name	nes
Number of rows	1397
Number of columns	4
Column type frequency:	
numeric	4
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
ObamaTherm	0	1	69.63	28.06	0	50	75	100	100	
RConserv	0	1	7.24	1.84	4	5	8	9	10	
ObamaConserv	0	1	4.98	1.72	3	4	4	6	9	
GOP	0	1	0.24	0.43	0	0	0	0	1	

SLM (simple linear model)

```r
obama_base <- lm(ObamaTherm ~ RConserv + GOP,
                 data = nes); tidy(obama_base)
```

```
## # A tibble: 3 x 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)   106.       2.60       40.6 2.19e-238
## 2 RConserv       -4.10     0.368     -11.2 9.48e- 28
## 3 GOP           -26.5      1.59      -16.7 2.82e- 57
```

Estimating models with multiplicative interactions

Expectation: Varying effects between ideology and party affiliation, with more or less extreme effects of ideology across its range and across party on feelings toward Obama,

$$\begin{aligned}\text{Obama thermometer} = \beta_0 + \beta_1(\text{Respondent conservatism}) + \beta_2(\text{GOP respondent}) \\ + \beta_3(\text{Respondent conservatism})(\text{GOP respondent}) + \epsilon\end{aligned}$$

Fit the model.

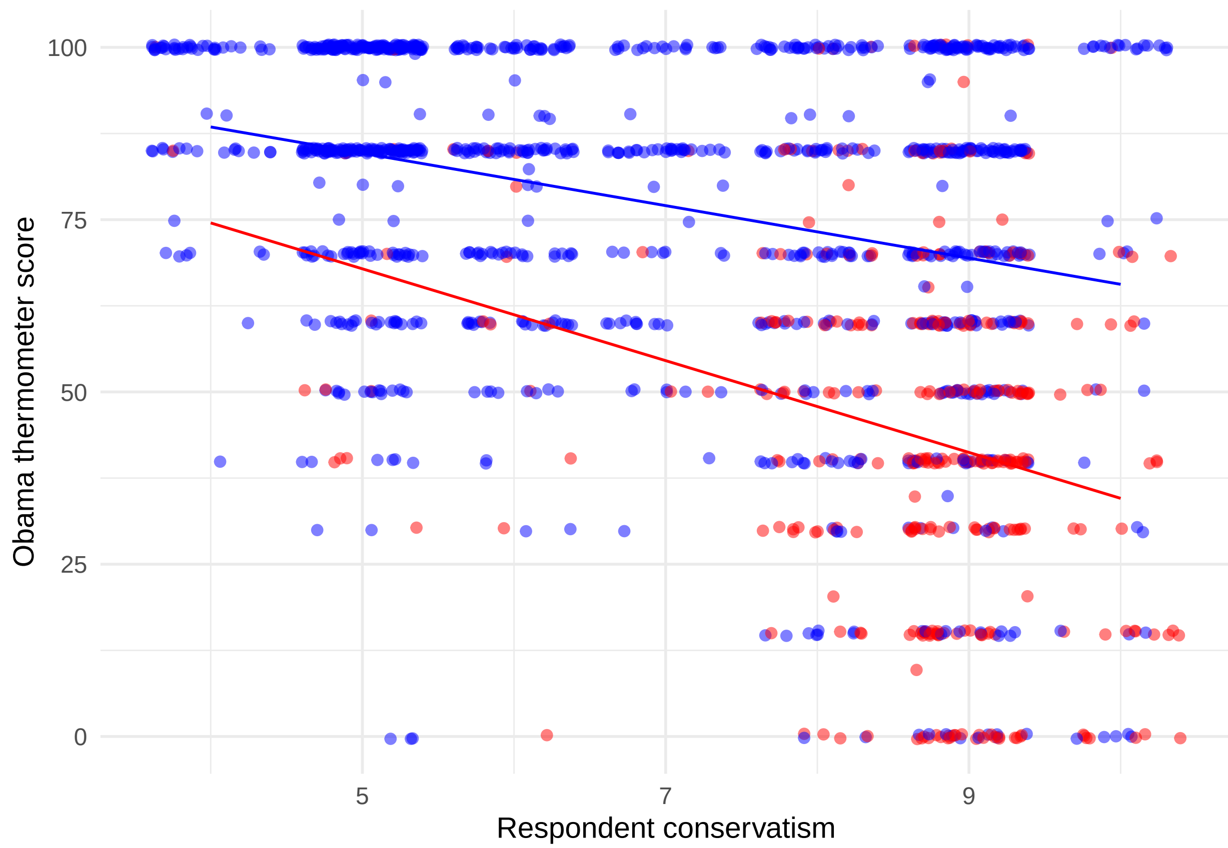
```r
obama_ideo_gop <- lm(ObamaTherm ~ RConserv * GOP, data = nes); tidy(obama_ideo_gop)
```

```
## # A tibble: 4 x 5
##   term         estimate std.error statistic   p.value
##   <chr>           <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)    104.       2.74      37.9   2.03e-216
```

```
## 2 RConserv        -3.81     0.388    -9.81  5.26e- 22
## 3 GOP             -2.50    10.2      -0.245 8.07e-  1
## 4 RConserv:GOP    -2.86     1.20     -2.38  1.75e-  2
```

Now, plot.

```
nes %>%
  add_predictions(obama_ideo_gop) %>%
  ggplot(aes(RConserv, ObamaTherm, color = factor(GOP))) +
  geom_jitter(alpha = .5) +
  geom_line(aes(y = pred)) +
  scale_color_manual(values = c("blue", "red")) +
  labs(x = "Respondent conservatism",
       y = "Obama thermometer score") +
  theme(legend.position = "none")
```



Another approach.

```
tidy(lm(ObamaTherm ~ RConserv, data = filter(nes, GOP == 0)))
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)   104.       2.67       38.9 3.23e-206
## 2 RConserv       -3.81     0.378     -10.1 7.87e- 23
```

```
tidy(lm(ObamaTherm ~ RConserv, data = filter(nes, GOP == 1)))
```

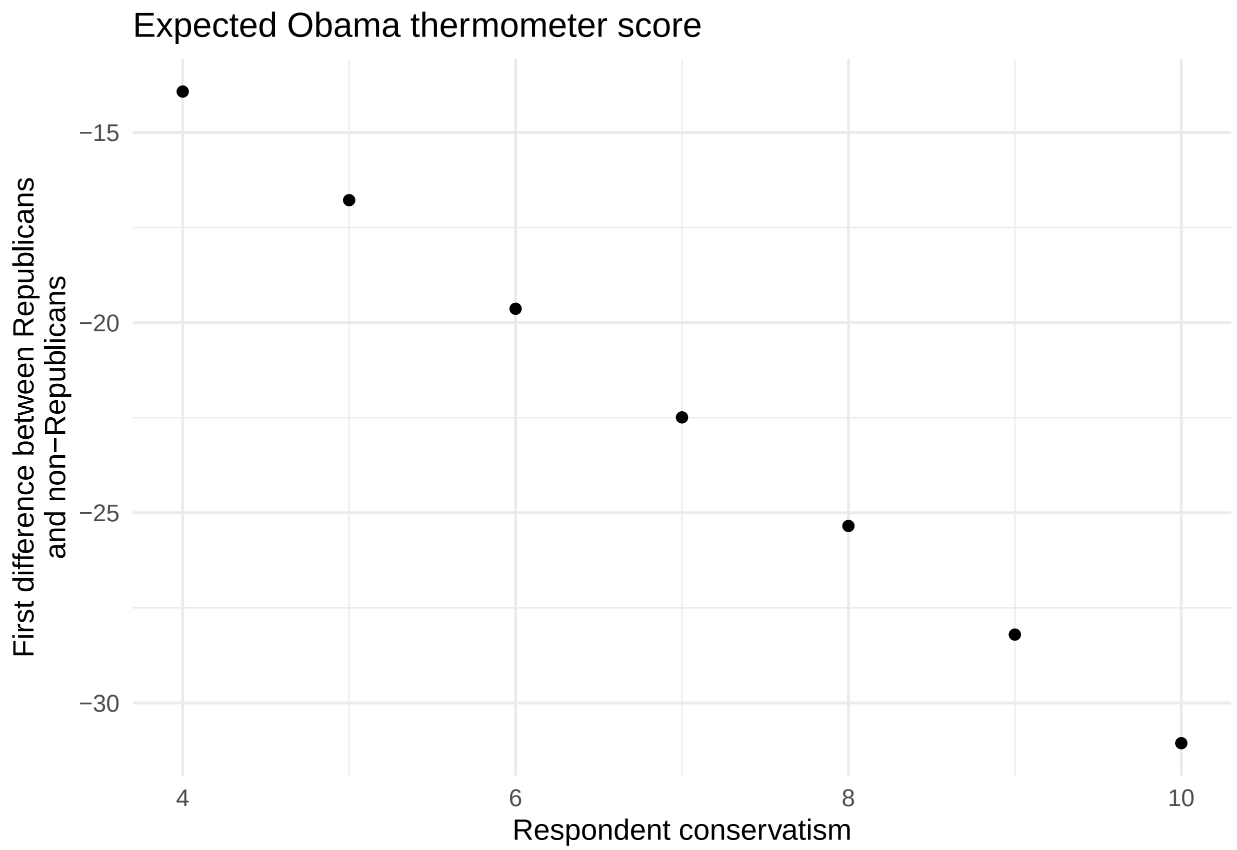
```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
```

```
##   <chr>            <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   101.        10.6       9.53 3.33e-19
## 2 RConserv       -6.66       1.22     -5.44 1.04e- 7
```

Causal direction

Exploring the first difference over party affiliation.

```
nes %>%
  data_grid(RConserv, GOP) %>%
  add_predictions(obama_ideo_gop) %>%
  spread(GOP, pred) %>%
  mutate(diff = `1` - `0`) %>%
  ggplot(aes(RConserv, diff)) +
  geom_point() +
  labs(title = "Expected Obama thermometer score",
       x = "Respondent conservatism",
       y = "First difference between Republicans\nand non-Republicans")
```



On your own

For this section, you will work in small groups of 4-5. *I will create these groups at random.*

IMPORTANT: *Don't forget that this code you're working on here is due at the appropriate Canvas module (in the form of an attachment to a "Discussion" post) prior to 5:00 pm CDT tomorrow. You need only submit a **single** file/script to be considered for credit (i.e., this .Rmd with your code inserted below each question). Recall, I don't care whether you got things right. I only care that attempts to each question have* *been made.*

Biden feelings data from the ANES. Load with the following code.

```r
library(tidyverse)
library(broom)
library(here)

biden <- read_csv(here("data", "biden.csv"))
```

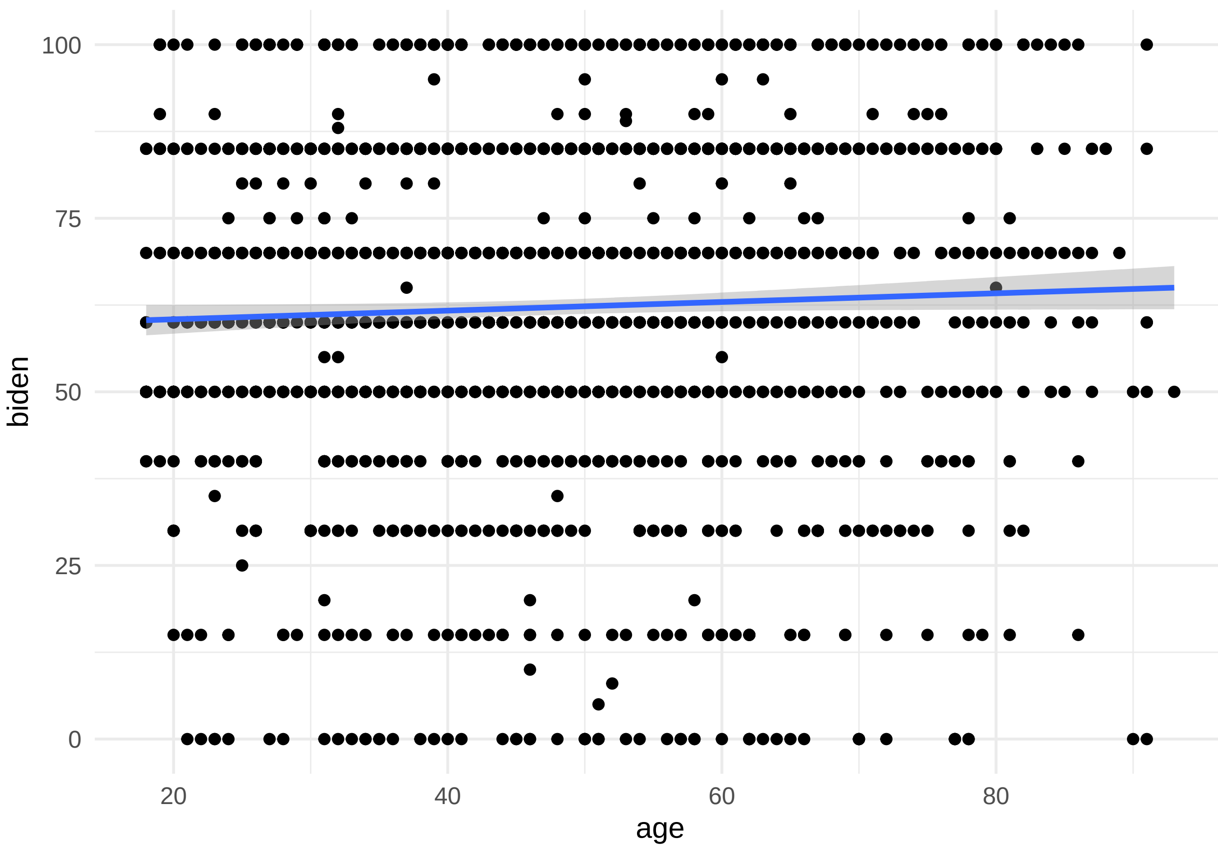
1. Estimate a **linear** model of the relationship between age (`age`) and attitudes toward Biden (`biden`), and plot the results. Remember to show the 95% confidence interval around your estimated fit line (hint: `geom_smooth()`). For reference, this simple model takes the form,

$$\text{Biden}_i = \beta_0 + \beta_1 \text{Age}$$

```r
biden_age_lm <- lm(biden ~ poly(age, 1, raw = TRUE), data = biden); tidy(biden_age_lm)
```

```
## # A tibble: 2 x 5
##   term                       estimate std.error statistic   p.value
##   <chr>                         <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)                 59.2       1.65       35.9  1.15e-213
## 2 poly(age, 1, raw = TRUE)    0.0624    0.0327      1.91 5.63e-  2
```

```r
ggplot(biden, aes(age, biden)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_minimal()
```



2. Relax the linear assumption to attempt to account for the fewer observations at the extreme values of

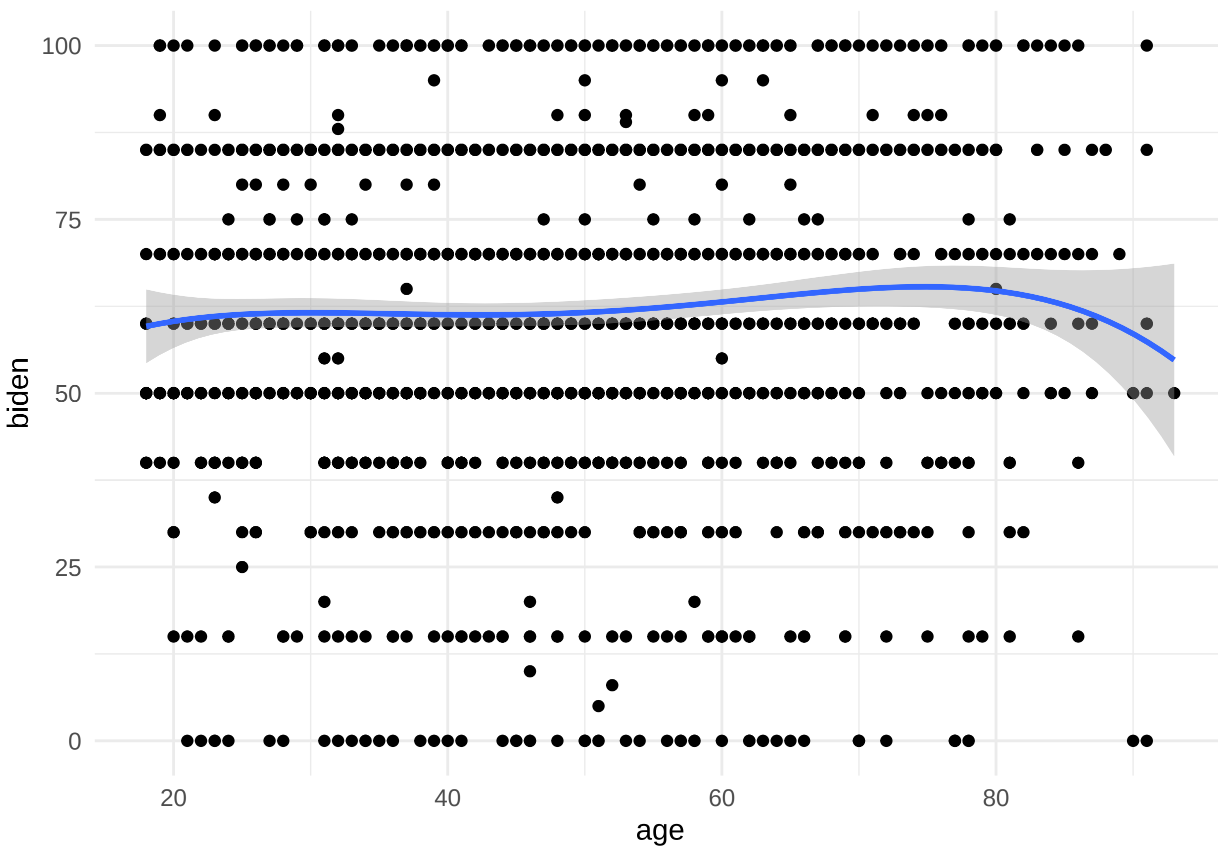
`age`, and estimate a fourth-order **polynomial** regression of the relationship between age and attitudes towards Biden (that is, wrap X's in `poly()`), and plot the results. Again, remember to show the 95% confidence interval around your estimated fit line. For reference, the fourth-order polynomial model takes the form,

$$\text{Biden}_i = \beta_0 + \beta_1 \text{Age} + \beta_2 \text{Age}^2 + \beta_3 \text{Age}^3 + \beta_4 \text{Age}^4$$

```
biden_4_order_lm <- lm(biden ~ poly(age, 4, raw = TRUE), data = biden); tidy(biden_4_order_lm)
```

```
## # A tibble: 5 x 5
##   term                         estimate   std.error statistic p.value
##   <chr>                           <dbl>       <dbl>     <dbl>   <dbl>
## 1 (Intercept)                  37.5        24.9          1.50   0.133
## 2 poly(age, 4, raw = TRUE)1   2.37         2.28          1.04   0.299
## 3 poly(age, 4, raw = TRUE)2 -0.0833       0.0730        -1.14   0.254
## 4 poly(age, 4, raw = TRUE)3  0.00122      0.000976       1.25   0.211
## 5 poly(age, 4, raw = TRUE)4 -0.00000622  0.00000464     -1.34   0.180
```

```
ggplot(biden, aes(age, biden)) +
  geom_point() +
  geom_smooth(method = "lm",
              formula = y ~ poly(x, 4, raw = TRUE))
```



3. In the figure produced in response to the previous question, you plotted the predicted values with the 95% confidence interval. In the case of ordinary linear regression (both the simple and polynomial models in our case), this is easy to estimate. Recall, the **standard error** is a measure of variance for the estimated parameter and is calculated by taking the square root (`sqrt()`) of the diagonal (`diag()`) of the variance-covariance matrix (`vcov()`). Standard errors, which are simply measures of uncertainty around some estimate, are critical to a traditional understanding of "statistical significance," which is reached by diagnosing t-statistics. Of note, t-statistics can be calculated by dividing estimated

coefficients (`$coefficients`) by their standard errors. So assuming errors are t-distributed, if these values are greater than 1.96 (the so-called "t-critical value for 95% confidence"), then the estimate is assumed to be "significant" at the 95% confidence level (note: $t >\approx 2.5$ for significance at 99% level).

- Obtain the variance-covariance matrix for your polynomial regression, and then calculate (by hand/don't call values from `broom::tidy()`) and report the standard errors for each parameter from your polynomial regression (*to answer this, make sure you read the question carefully*).

```
SEs <-  sqrt(diag(vcov(biden_4_order_lm)))
SEs
```

```
##              (Intercept) poly(age, 4, raw = TRUE)1 poly(age, 4, raw = TRUE)2
##             2.489986e+01              2.282029e+00              7.303672e-02
## poly(age, 4, raw = TRUE)3 poly(age, 4, raw = TRUE)4
##             9.762763e-04              4.635816e-06
```

```
* Calculate (by hand/don't call values from `broom::tidy()`) and report the t-statistics for each of pa
```

```
t_statistics_calculation <- function(position){
  biden_4_order_lm$coefficients[position] / SEs[position]
}

intercept_order_t_statistics <- t_statistics_calculation(1)
intercept_order_t_statistics
```

```
## (Intercept)
##    1.504289
```

```
first_order_t_statistics <- t_statistics_calculation(2)
first_order_t_statistics
```

```
## poly(age, 4, raw = TRUE)1
##                  1.038675
```

```
second_order_t_statistics <- t_statistics_calculation(3)
second_order_t_statistics
```

```
## poly(age, 4, raw = TRUE)2
##                 -1.140109
```

```
third_order_t_statistics <- t_statistics_calculation(4)
third_order_t_statistics
```

```
## poly(age, 4, raw = TRUE)3
##                  1.251407
```

```
fourth_order_t_statistics <- t_statistics_calculation(5)
fourth_order_t_statistics
```

```
## poly(age, 4, raw = TRUE)4
##                 -1.342676
```

```
* Which coefficient estimates are significant at the 95% level, and which are not? *Hint:* you might co
```

```
broom::tidy(biden_4_order_lm) # According to the result, my solution given above is correct
```

```
## # A tibble: 5 x 5
##   term                        estimate   std.error statistic p.value
##   <chr>                          <dbl>       <dbl>     <dbl>   <dbl>
## 1 (Intercept)               37.5        24.9            1.50   0.133
## 2 poly(age, 4, raw = TRUE)1  2.37        2.28           1.04   0.299
```

```
## 3 poly(age, 4, raw = TRUE)2 -0.0833      0.0730         -1.14   0.254
## 4 poly(age, 4, raw = TRUE)3  0.00122     0.000976        1.25   0.211
## 5 poly(age, 4, raw = TRUE)4 -0.00000622  0.00000464     -1.34   0.180
```

As we know, if these values are greater than 1.96 (the so-called "t-critical value for 95% confidence"), then the estimate is assumed to be "significant" at the 95% confidence level. All the t values are less than 1.96. Therefore, none of the coefficients are significant at the 95% level.

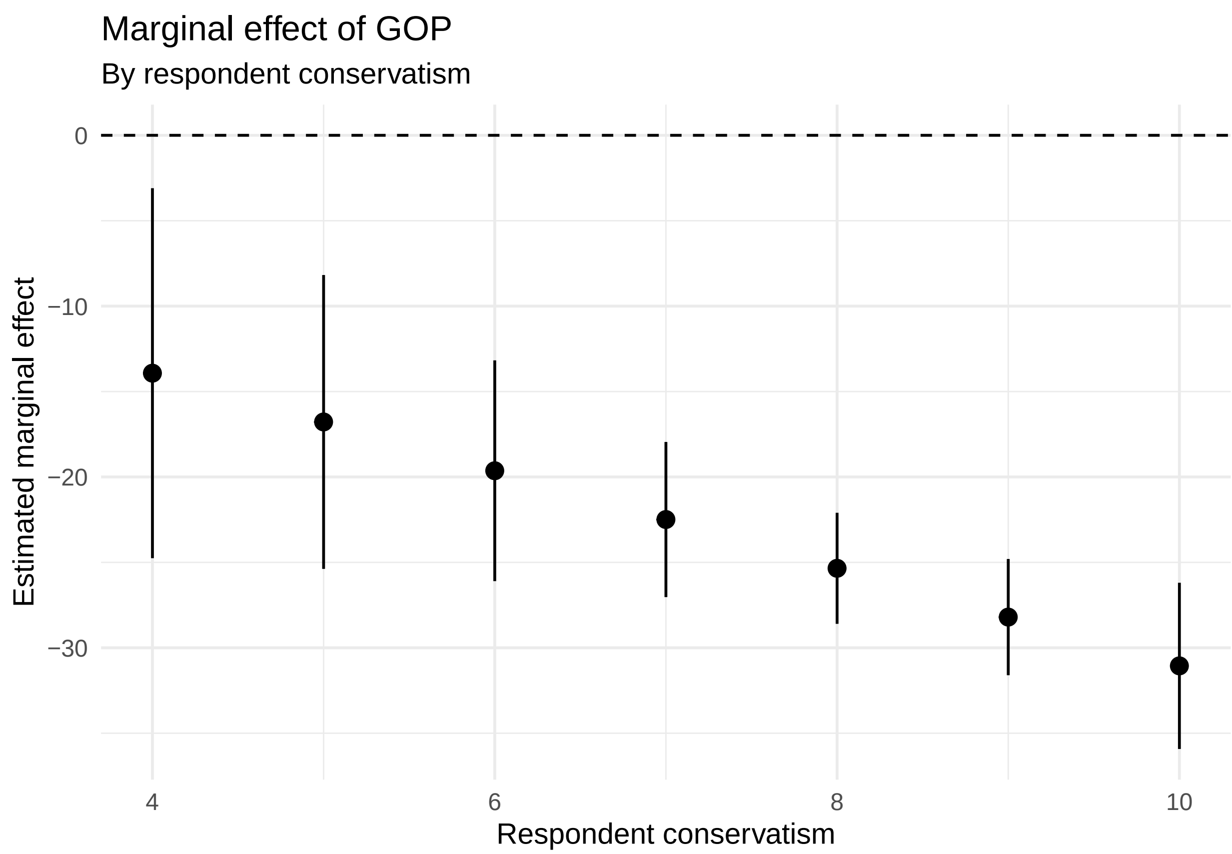
This substantively means the model is not effective at all. No terms (variables) have any effect on the Y variable. One's age is not a influential factor affecting the attitude toward Biden.

Appendix: Some extra code for the interested user

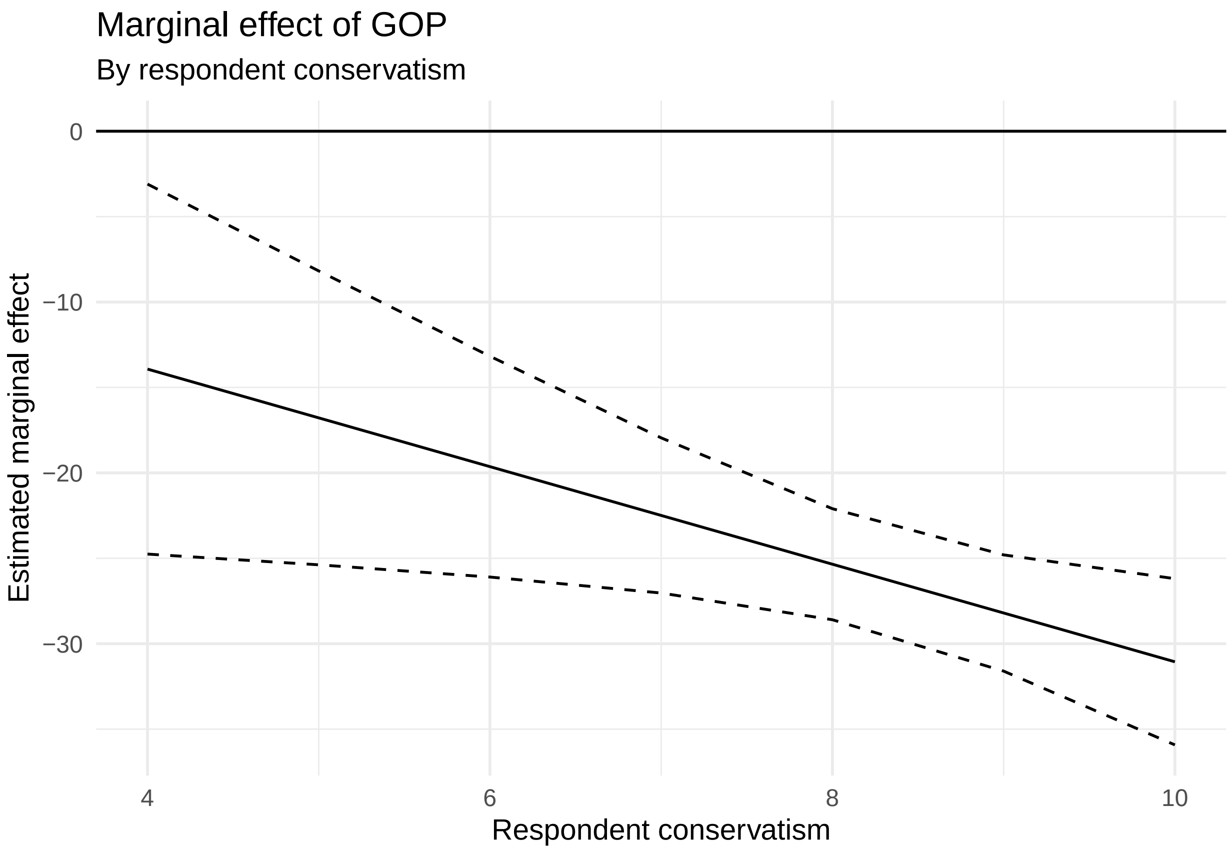
If we want to conduct inference on $\hat{\psi}_1$ or $\hat{\psi}_2$ (the marginal effect of either X on Y), we can do that as well:

```
# function to get coefficient estimates and standard errors
# model -> lm object
# mod_var -> name of moderating variable in the interaction

instant_effect <- function(model, mod_var){
  # get interaction term name
  int.name <- names(model$coefficients)[[which(str_detect(names(model$coefficients), ":"))]]

  marg_var <- str_split(int.name, ":")[[1]][[which(str_split(int.name, ":")[[1]] != mod_var)]]

  # store coefficients and covariance matrix
  beta.hat <- coef(model)
  cov <- vcov(model)

  # possible set of values for mod_var
  if(class(model)[[1]] == "lm"){
    z <- seq(min(model$model[[mod_var]]), max(model$model[[mod_var]]))
  } else {
    z <- seq(min(model$data[[mod_var]]), max(model$data[[mod_var]]))
  }

  # calculate instantaneous effect
  dy.dx <- beta.hat[[marg_var]] + beta.hat[[int.name]] * z

  # calculate standard errors for instantaeous effect
  se.dy.dx <- sqrt(cov[marg_var, marg_var] +
                   z^2 * cov[int.name, int.name] +
                   2 * z * cov[marg_var, int.name])

  # combine into data frame
  tibble(z = z,
         dy.dx = dy.dx,
         se = se.dy.dx)
}

# point range plot
instant_effect(obama_ideo_gop, "RConserv") %>%
  ggplot(aes(z, dy.dx,
             ymin = dy.dx - 1.96 * se,
             ymax = dy.dx + 1.96 * se)) +
```

```r
  geom_pointrange() +
  geom_hline(yintercept = 0, linetype = 2) +
  labs(title = "Marginal effect of GOP",
       subtitle = "By respondent conservatism",
       x = "Respondent conservatism",
       y = "Estimated marginal effect")
```




```r
# line plot
instant_effect(obama_ideo_gop, "RConserv") %>%
  ggplot(aes(z, dy.dx)) +
  geom_line() +
  geom_line(aes(y = dy.dx - 1.96 * se), linetype = 2) +
  geom_line(aes(y = dy.dx + 1.96 * se), linetype = 2) +
  geom_hline(yintercept = 0) +
  labs(title = "Marginal effect of GOP",
       subtitle = "By respondent conservatism",
       x = "Respondent conservatism",
       y = "Estimated marginal effect")
```


Marginal effect of GOP
By respondent conservatism
Estimated marginal effect
0
−10
−20
−30
4
6
8
10
Respondent conservatism