Week 6 Lab

Confidence Intervals and Hypothesis Testing

Micaela Wood
02/10/2022

Today

- Review from first half of course
- Learn how to manually create, plot, and interpret confidence intervals
- Learn how to conduct hypothesis tests and use **if** statements

Review

- how to use the mutate function
- how to plot from separate data frames
- how to plot using functions

Mutate

For this part I will be going through question 1 from the second homework set

```r
library(tidyverse)

n <- 100
set.seed(12345)
# Generate data in a tibble
data = tibble(
  e = rnorm(n, sd = 30),
  v = rnorm(n, sd = 20),
  x = runif(n, min = 0, max = 10),
  y = 8 - 3*x + e,
  z = 20 - 2*y + v,
)
```

Mutate

For this problem we needed to create standardized versions of the variables

```r
data_std <- data %>% mutate(
  ??? = ???,
  ??? = ???
)
```

Mutate

For this problem we needed to create standardized versions of the variables

```
data_std <- data %>% mutate(
  x.std = (x - mean(x))/sd(x),
  y.std = (y - mean(y))/sd(y)
)
```

Mutate

Another Option

```r
x_mu = mean(data$x)
y_mu = mean(data$y)

x_sd = std(data$x)
y_sd = std(data$y)

data_std <- data %>% mutate(
  x.std = (x - x_mu)/x_sd,
  y.std = (y - y_mu)/y_sd
)
```

Plots from different dataframes

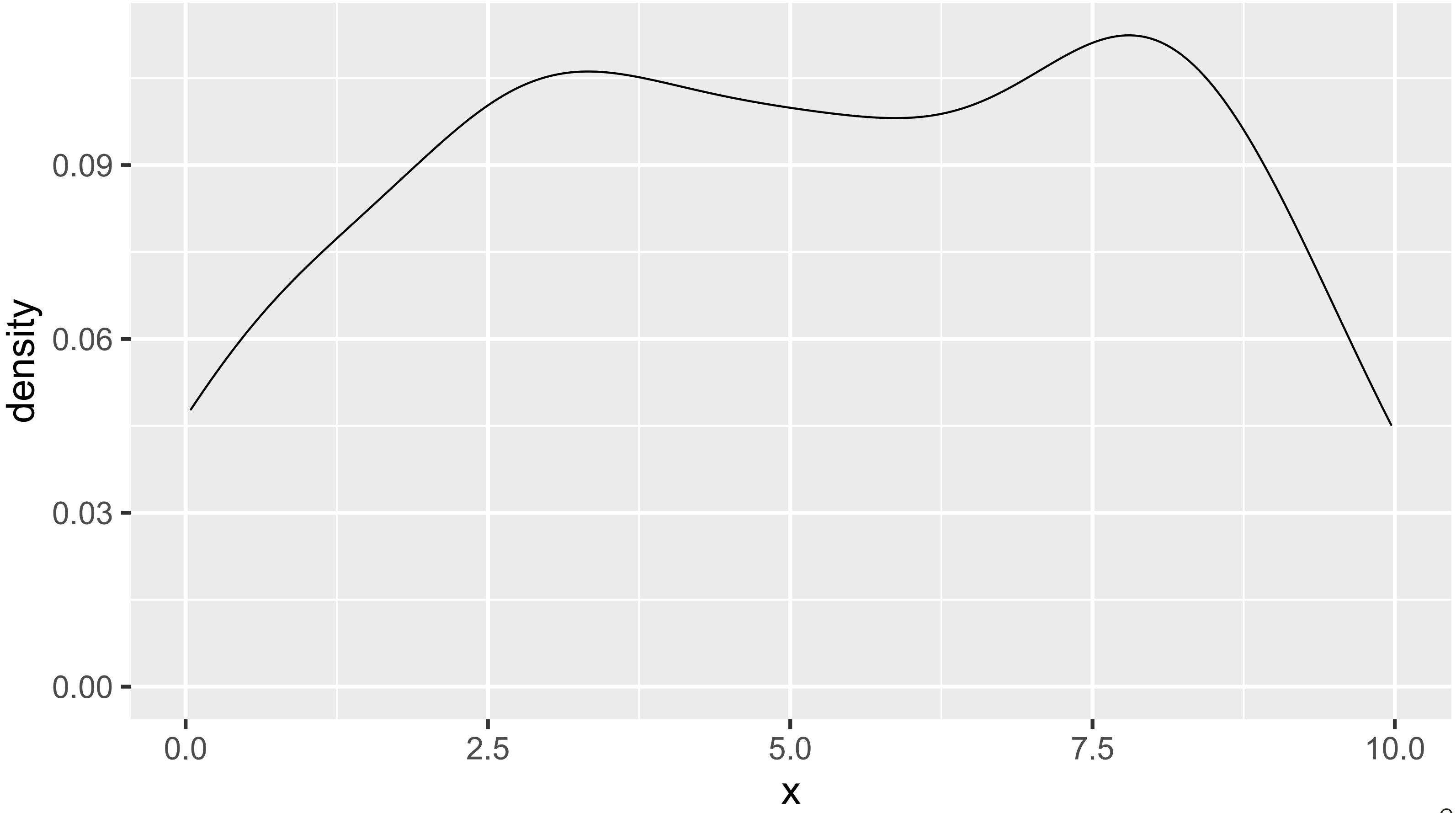
For the last part of this problem we needed to make density plots of the regular and standardized x's

First let's just plot one density plot

```
ggplot()+
  ???(data = ???, aes(??? = ???))
```

Plots from different dataframes

```
ggplot()+
  geom_density(data = data, aes(x = x))
```



Plots from different dataframes

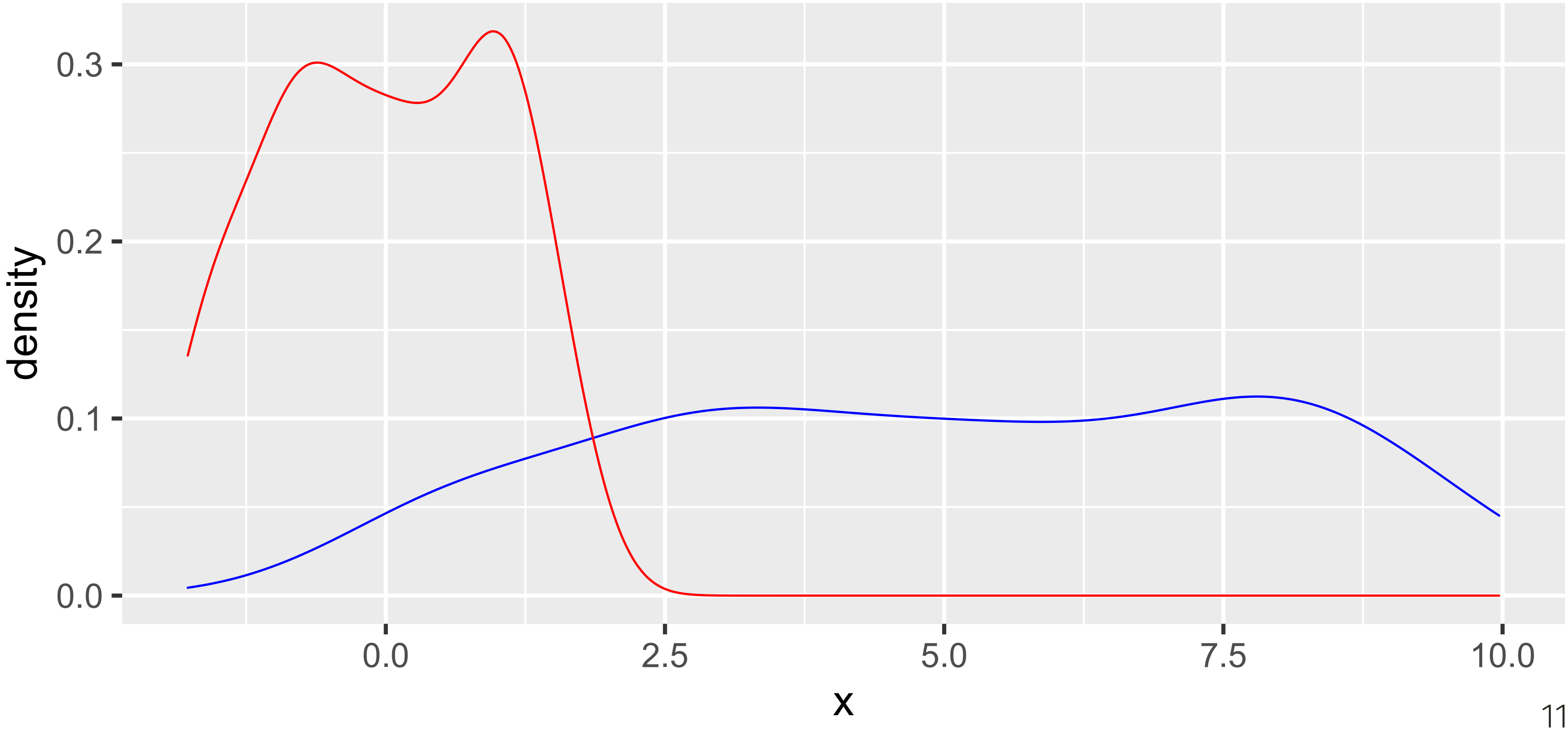
Now we can try to make a second plot on top of the first

```
ggplot()+
  geom_density(data = data, aes(x = x))+
  ???(data = ???, aes(x = ???))
```

Plots from different dataframes

I will also add some color to this one

```
ggplot()+
  geom_density(data = data, aes(x = x), color = "blue")+
  geom_density(data = data_std, aes(x = x.std), color = "red")
```



Plots of conditional means

Now I will go over the Plot from Question 1 on homework 3

I will first load the data

```r
nlsy79 <- read_csv("nlsy79.csv")
```

Plots of conditional means

There are 2 ways to get the desired outcome:

1. Create a data frame that is grouped by years of schooling

2. Put a function inside of ggplot

Plots of conditional means

1. Create a data frame that is grouped by years of schooling

```
nlsy79_summ <- nlsy79 %>% group_by(hgc) %>%
  summarize(mean_earn = mean(earn2009, na.rm=TRUE)) %>%
  filter(is.na(hgc)==0)

head(nlsy79_summ)
```

```
#> # A tibble: 6 x 2
#>     hgc mean_earn
#>   <dbl>     <dbl>
#> 1     0    12150.
#> 2     1         0
#> 3     3    19882.
#> 4     5         0
#> 5     6    17673.
#> 6     7     9770.
```

Plots of conditional means

There are three functions here that we may not be familiar with

- `group_by()`:
  - This tells R how we want the data set to be grouped
  - Because we wanted means conditional on year of schooling we did `group_by(hgc)`
- `summarize`:
  - This is similar to mutate
  - It creates variables for our grouped data frame
- `filter`:
  - We have used this before but not often

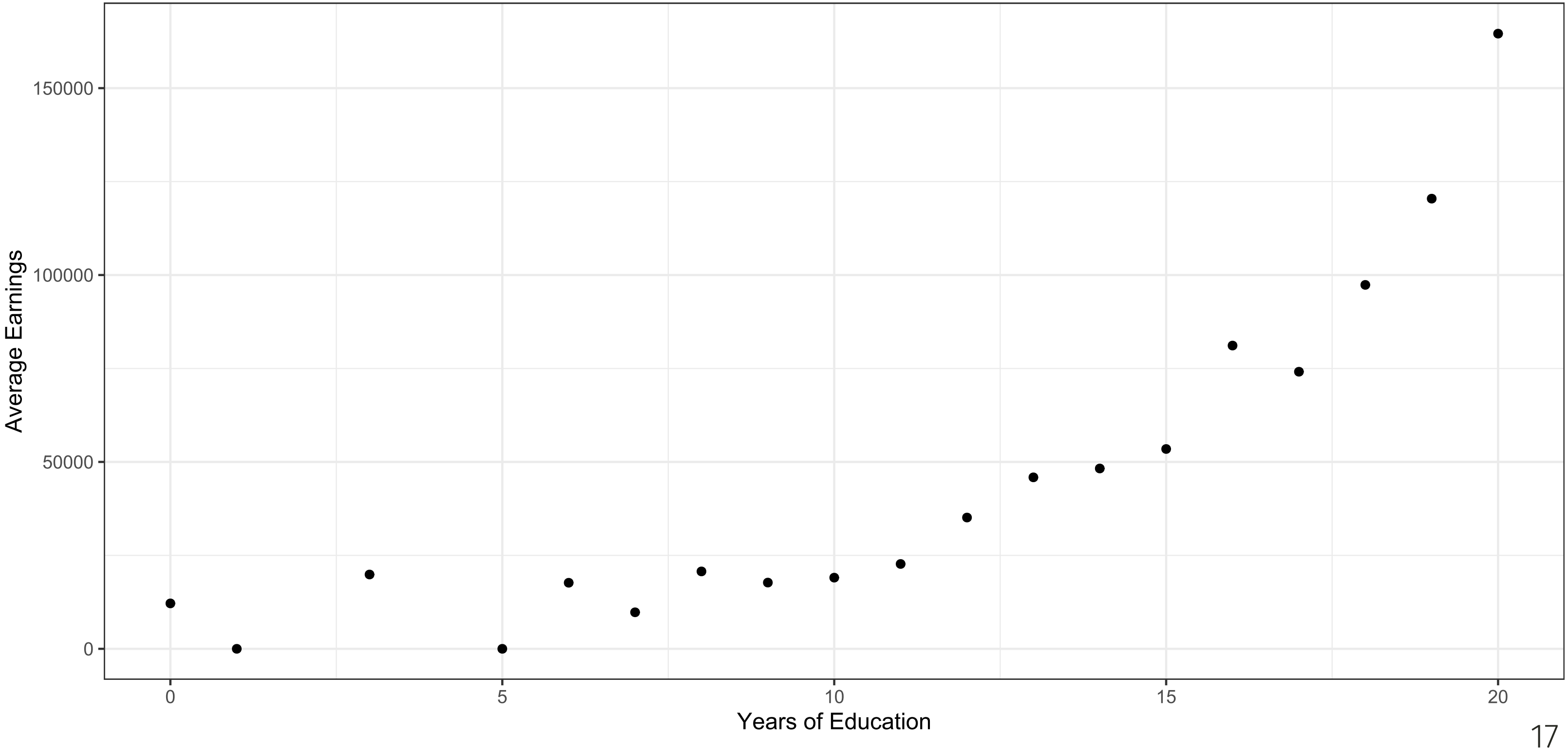
Plots of conditional means

Now that we have a grouped data frame we can make a plot.

```
ggplot(data = ???) +
  geom_point(aes(x = ???, y = ???))+
  ylab("???") +
  xlab("???") +
  theme_bw()
```

Plots of conditional means

```
ggplot(nlsy79_summ) +
  geom_point(aes(hgc, mean_earn))+
  ylab("Average Earnings") +
  xlab("Years of Education") +
  theme_bw()
```



Plots of conditional means

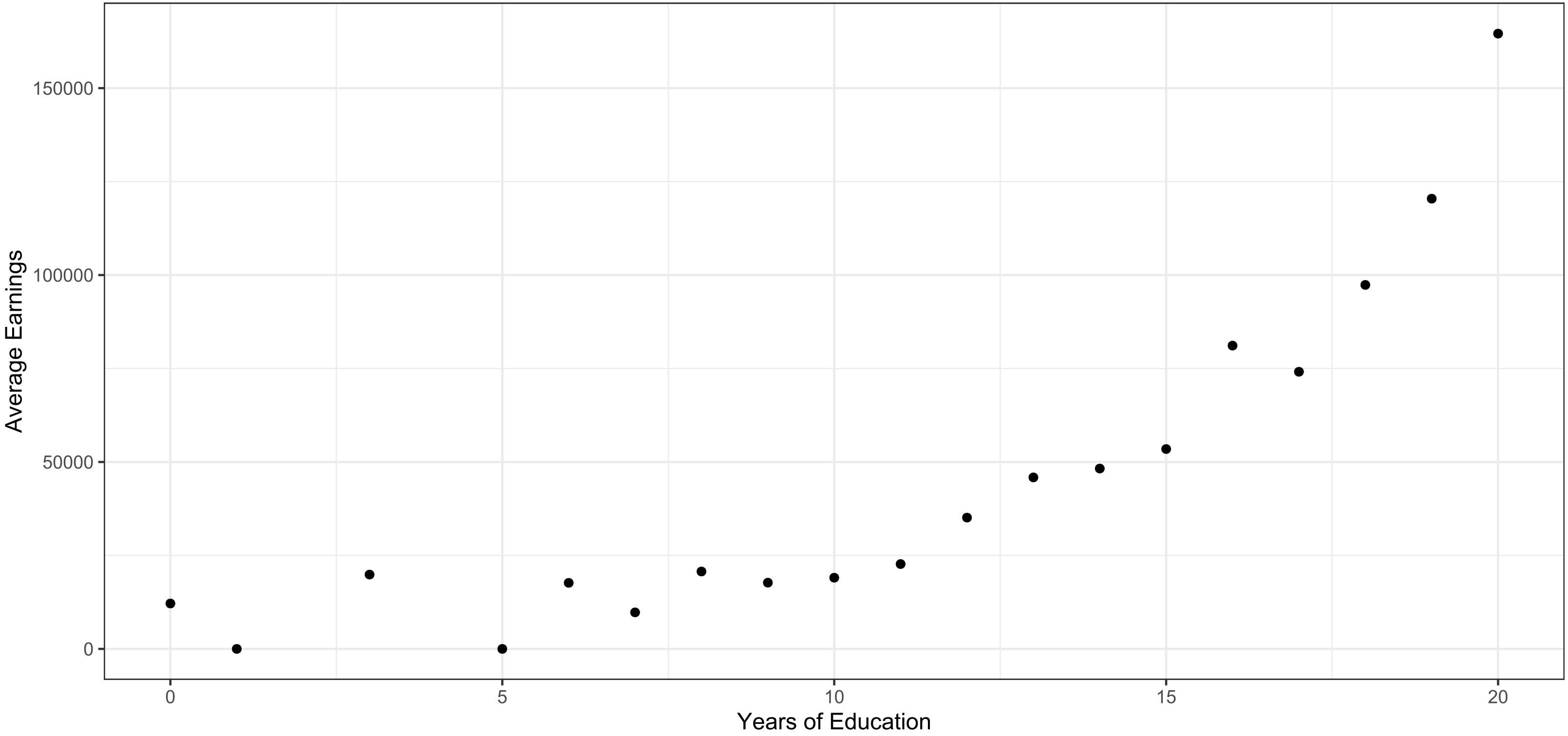
1. Put a function inside ggplot
    - Don't forget we can use `help` for new functions

```r
ggplot(data = ??? ,aes(x=???,y=???)) +
  stat_summary(fun = "???", geom = "???")+
  xlab("Years of Education") + ylab("Average Earnings") +
  theme_bw()
```

- `stat_summary` allows us to summarise y based on groups of x
- `fun =` is how we want the y value summarized.
- `geom =` gives us the style of graph that we want.

Plots of conditional means

```
ggplot(nlsy79,aes(x=hgc,y=earn2009)) +
  stat_summary(fun = "mean", geom = "point")+
  xlab("Years of Education") + ylab("Average Earnings") +
  theme_bw()
```



Questions?

Confidence Intervals

Things we need to calculate confidence intervals

- Estimate of $\hat{\beta}_2$
- Estimate of Standard Errors
- t Statistic

With these we can calculate...

- Upper confidence bound
- Lower confidence bound

Confidence Intervals

We can get everything we need to calculate a confidence interval from our regression output

For this I will use the data from homework 3

```r
n <- 1000
set.seed(1245)

data_sim = tibble(
  e = rnorm(n, sd = 30),
  v = rnorm(n, sd = 20),
  x = runif(n, min = 0, max = 10),
  y = 8 - 3*x + e,
  z = 20 - 0.3*y + 3*x + v
)
```

Confidence Intervals

Let's run the regressions from that homework

```
lm1 ← lm(data = ???, ???)
lm2 ← lm(data = ???, ???)
```

Confidence Intervals

Let's run the regressions from that homework

```
lm1 ← lm(data = data_sim, z~y)
lm2 ← lm(data = data_sim, z~y+x)
```

Confidence Intervals

We can save these results from the regression in an easy to use manner with `tidy()`

```
results <- rbind(tidy(???), tidy(???)) %>% filter(term =="???")

results
```

Confidence Intervals

We can save these results from the regression in an easy to use manner with `tidy()`

```
results <- rbind(tidy(lm1), tidy(lm2)) %>% filter(term =="y")

results
```

```
#> # A tibble: 2 x 5
#>   term  estimate std.error statistic  p.value
#>   <chr>    <dbl>     <dbl>     <dbl>    <dbl>
#> 1 y       -0.355    0.0227     -15.6 2.17e-49
#> 2 y       -0.273    0.0214     -12.8 9.69e-35
```

Notice that this is a dataframe with columns for our estimate and standard errors

Confidence Intervals

Now we can use `mutate` to create variables for our confidence interval

```r
results <- results %>% mutate(
  upper_bound = ???,
  lower_bound = ???,
  model = c("???", "???")
)
```

Confidence Intervals

Now we can use `mutate` to create variables for our confidence interval

```
results ← results %>% mutate(
  lower_bound = estimate - 1.96*std.error,
  upper_bound = estimate + 1.96*std.error,
  model = c("without x", "with x")
)

results
```

```
#> # A tibble: 2 x 8
#>   term  estimate std.error statistic  p.value lower_bound upper_bound model
#>   <chr>    <dbl>     <dbl>     <dbl>    <dbl>       <dbl>       <dbl> <chr>
#> 1 y       -0.355    0.0227     -15.6 2.17e-49      -0.400      -0.311 without x
#> 2 y       -0.273    0.0214     -12.8 9.69e-35      -0.315      -0.231 with x
```

Confidence Intervals

We may also be interested in plotting our confidence intervals

To do this we can use `geom_errorbar`

`geom_errorbar` requires a ymax and ymin.

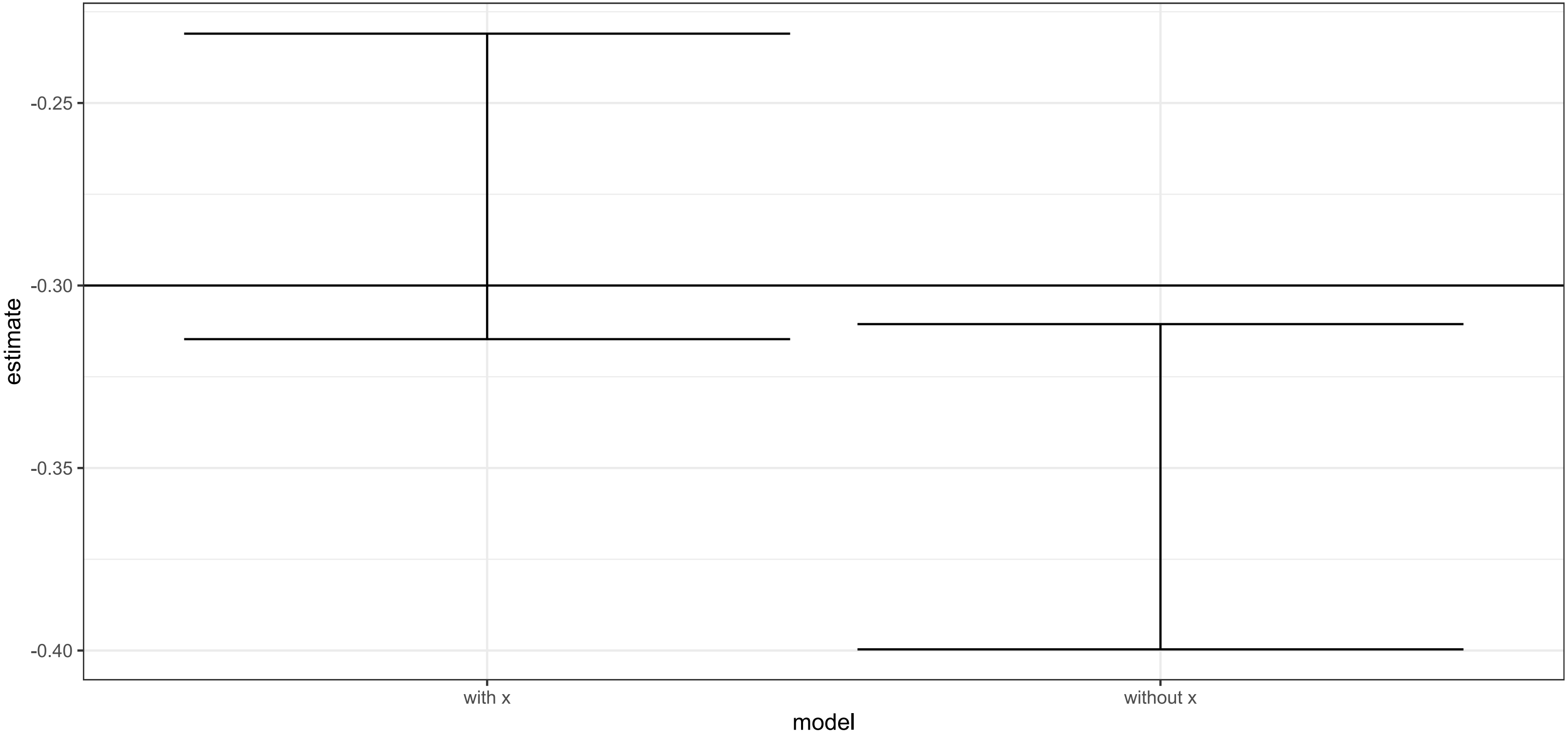
- These are the upper and lower bounds for our confidence intervals

Confidence Intervals

```r
ggplot(data = ???, aes(x=???, y=???))+
  geom_errorbar(aes(ymin=???, ymax=???))+
  geom_hline(yintercept=???)+
  theme_bw()
```

Confidence Intervals

```
ggplot(data = results, aes(x=model, y=estimate))+
  geom_errorbar(aes(ymin=lower_bound, ymax=upper_bound))+
  geom_hline(yintercept=-0.3)+
  theme_bw()
```



Hypothesis Testing

For Hypothesis Testing we can use the same results data frame

```
results
```

```
#> # A tibble: 2 x 8
#>   term  estimate std.error statistic  p.value lower_bound upper_bound model
#>   <chr>    <dbl>     <dbl>     <dbl>    <dbl>       <dbl>       <dbl> <chr>
#> 1 y       -0.355    0.0227     -15.6 2.17e-49      -0.400      -0.311 without x
#> 2 y       -0.273    0.0214     -12.8 9.69e-35      -0.315      -0.231 with x
```

With this information we will can create a test statistic

Hypothesis Testing

The test we will conduct is:

$$H_o : \beta = \mathbf{-0.3}$$

$$H_a : \beta \neq \mathbf{-0.3}$$

Hypothesis Testing

To create our test statistic I am going to pull the data out of the data frame.

First I will test the model without x

```
#pull out beta
beta1 = results$???[???]

#pull out standard error
st.err1 = results$???[???]
```

Hypothesis Testing

To create our test statistic I am going to pull the data out of the data frame.

First I will test the model without x

```
#pull out beta
beta1 = results$estimate[1]

#pull out standard error
st.err1 = results$std.error[1]
```

Hypothesis Testing

The test we will conduct is:

$$H_o : \beta = -0.3$$

$$H_a : \beta \neq -0.3$$

Now I will create the test statistic

```
test1 = (??? - ???)/???
```

Hypothesis Testing

The test we will conduct is:

$$H_o : \beta = -0.3$$

$$H_a : \beta \neq -0.3$$

Now I will create the test statistic

```
test1 = (beta1 - (-0.3))/st.err1
test1
```

```
#> [1] -2.425387
```

Do we reject or fail to reject the null hypothesis?

|-2.425| > 1.96 so **reject** the null hypothesis

Hypothesis Testing

Now lets repeat this for the model with x

Let's pull the numbers we need from this model

```
#pull out beta
beta2 = results$estimate[???]

#pull out standard error
st.err2 = results$std.error[???]
```

Hypothesis Testing

Now lets repeat this for the model with x

Let's pull the numbers we need from this model

```
#pull out beta
beta2 = results$estimate[2]

#pull out standard error
st.err2 = results$std.error[2]
```

Hypothesis Testing

Now we repeat the hypothesis test

$$H_o : \beta = -0.3$$

$$H_a : \beta \neq -0.3$$

I will create the new test statistic

```
test2 = (beta2 - (-0.3))/st.err2
test2
```

```
#> [1] 1.271542
```

Do we reject or fail to reject the null hypothesis?

|1.27| < 1.96 so **fail to reject** the null

Bonus

If Statements

- We can create an `if` statement to print out the results of our hypothesis tests
- I prefer to use `ifelse`

```
ifelse(test, if true, if false)
```

If Statements

Lets start with a simple example.

```
ifelse(test, if true, if false)
```

```
ifelse(2 + 2 == 4, "This is True", "This is False")
```

```
#> [1] "This is True"
```

```
ifelse(2 + 2 > 5, "This is True", "This is False")
```

```
#> [1] "This is False"
```

If Statements

Let's try to make one for our first Hypothesis tests

```
ifelse(test, if true, if false)
```

```
ifelse(abs(test1) > 1.96,
       "Reject the null hypothesis",
       "Fail to reject the null hypothesis")
```

```
#> [1] "Reject the null hypothesis"
```

If Statements

Now we can do the same thing for the second test

```
ifelse(test, if true, if false)
```

```
ifelse(abs(test2) > 1.96,
       "Reject the null hypothesis",
       "Fail to reject the null hypothesis")
```

```
#> [1] "Fail to reject the null hypothesis"
```

Questions?