

Regression modelling with a categorical exposure

Corinne Riddell

December 2, 2020

Learning objectives for today

- Learn how to generalize the simple regression model to the case when the x variable is categorical (and y is still continuous)
- Learn how to run these models in R

Example

Calcium is an essential mineral that regulates the heart, is important for blood clotting and for building healthy bones. The National Osteoporosis Foundation recommends a daily calcium intake of 1000-1200 mg/day for adult men and women. While calcium is contained in some foods, most adults do not get enough calcium in their diets and take supplements. Unfortunately some of the supplements have side effects such as gastric distress, making them difficult for some patients to take on a regular basis.

A study is designed to test whether there is a difference in mean daily calcium intake in adults with normal bone density, adults with osteopenia (a low bone density which may lead to osteoporosis) and adults with osteoporosis. Adults 60 years of age with normal bone density, osteopenia and osteoporosis are selected at random from hospital records and invited to participate in the study. Each participant's daily calcium intake is measured based on reported food intake and supplements. The data are shown below.

The data

```
#students, don't worry about knowing the next lines of code. Just have a look at  
#the dataset so you understand what variables it contains  
normal <- c(1200, 1000, 980, 900, 750, 800)  
osteopenia <- c(1000, 1100, 700, 800, 500, 700)  
osteoporosis <- c(890, 650, 1100, 900, 400, 350)
```

```
calcium_data <- data.frame(calcium_intake = c(normal, osteopenia, osteoporosis),  
                           type = c(rep("normal", 6), rep("osteopenia", 6),  
                                   rep("osteoporosis", 6)),  
                           type_num = c(rep(1, 6), rep(2, 6), rep(3, 6)))
```

calcium_data

##	calcium_intake	type	type_num
## 1	1200	normal	1
## 2	1000	normal	1
## 3	980	normal	1
## 4	900	normal	1
## 5	750	normal	1
## 6	800	normal	1
## 7	1000	osteopenia	2
## 8	1100	osteopenia	2

```
## 9          700  osteopenia      2
## 10         800  osteopenia      2
## 11         500  osteopenia      2
## 12         700  osteopenia      2
## 13         890 osteoporosis      3
## 14         650 osteoporosis      3
## 15        1100 osteoporosis      3
## 16         900 osteoporosis      3
## 17         400 osteoporosis      3
## 18         350 osteoporosis      3
```

Refresher on `lm()`

- So far, we’ve run linear regression when both the outcome and explanatory variables are continuous
- We can also use linear regression when the outcome is continuous, but the explanatory variable is categorical
- In today’s lecture we learn how to run and interpret these sorts of models and see how the results compare to the output from an ANOVA model.

Running linear regression using a categorical explanatory variable

The first thing you want to do is check how the categorical variable is coded and change the order of the categorical (factor variable) if you need to.

```
str(calcium_data)
```

```
## 'data.frame':  18 obs. of  3 variables:
## $ calcium_intake: num  1200 1000 980 900 750 800 1000 1100 700 800 ...
## $ type          : chr  "normal" "normal" "normal" "normal" ...
## $ type_num      : num   1 1 1 1 1 1 2 2 2 2 ...
```

This shows us that `type` is coded as “chr”, which stands for character. This means it isn’t yet stored as a factor variable, which is what we want to have for putting it into the analysis.

Making `type` a factor variable

This code updated the variable `type` to be stored as a factor variable. We overwrote the original variable, but you might want to rename it “`type2`” within `mutate` so you can compare the new and old variables if you’re doing this for your first time!

```
calcium_data <- calcium_data %>% mutate(type = factor(type))
str(calcium_data)
```

```
## 'data.frame':  18 obs. of  3 variables:
## $ calcium_intake: num  1200 1000 980 900 750 800 1000 1100 700 800 ...
## $ type          : Factor w/ 3 levels "normal","osteopenia",...: 1 1 1 1 1 1 2 2 2 2 ...
## $ type_num      : num   1 1 1 1 1 1 2 2 2 2 ...
```

Now we can see that `type` is a Factor variable with three levels. It is sorted alphabetically, which for our purposes is okay because we would like “normal”, the first category, to be the **referent** group.

Referent group: The group that the others will be compared to. Here, we will compare the osteopenia and osteoporosis groups to the normal group. Oftentimes, we set the referent group to be the level with the relatively best health outcome compared to the other groups, or the group with the largest sample size.

Defining a new referent group

If you wanted to change the referent group, you could use `fct_relevel()`, which we used in Part I of the course.

```
calcium_data <- calcium_data %>% mutate(type_reordered = fct_relevel(type, "osteoporosis")) #makes osteoporosis the referent group
levels(calcium_data$type_reordered)
```

```
## [1] "osteoporosis" "normal"      "osteopenia"
```

- We will run two regressions, one with `type` and the other with `type_reordered` to see how their outputs differ.

Linear regression when x is categorical

Recall the form of the regression model when x and y are both continuous:

$$y = a + bx$$

We can write this more precisely. The predicted value for individual i is represented by:

$$\hat{y}_i = \hat{a} + \hat{b}x_i$$

Suppose you have a categorical variable with three levels. Then the new form of the regression model is:

$$\hat{y}_i = \hat{a} + \hat{b}_1 * [i\text{'s category} == \text{osteopenia}] + \hat{b}_2[i\text{'s category} == \text{osteoporosis}]$$

Linear regression when x is categorical

General form of regression model for categorical x variable with three levels:

$$\hat{y}_i = \hat{a} + \hat{b}_1 * [i\text{'s category} == \text{osteopenia}] + \hat{b}_2[i\text{'s category} == \text{osteoporosis}]$$

- When an individual is in the normal bone density category, then both `category == osteopenia` and `category == osteoporosis` are FALSE and the regression model simplifies to:

$$\hat{y}_i = \hat{a}$$

- When an individual is in the osteopenia bone density category, then `category == osteopenia` is TRUE and `category == osteoporosis` is FALSE and the regression model simplifies to:

$$\hat{y}_i = \hat{a} + \hat{b}_1$$

- When an individual is in the osteoporosis bone density category, then `category == osteopenia` is FALSE and `category == osteoporosis` is TRUE and the regression model simplifies to:

$$\hat{y}_i = \hat{a} + \hat{b}_2$$

Thus, when x is categorical, the regression model predicts each individual's measure to be at the mean for that category.

Running `lm()` on categorical x data

```
library(broom)
calcium_lm <- lm(calcium_intake ~ type, data = calcium_data)

tidy(calcium_lm)

## # A tibble: 3 x 5
##   term          estimate std.error statistic    p.value
##   <chr>          <dbl>    <dbl>    <dbl>    <dbl>
## 1 (Intercept)      938.      95.4      9.83 0.0000000625
## 2 typeosteopenia  -138.     135.     -1.02 0.322
## 3 typeosteoporosis -223.     135.     -1.65 0.119
```

Interpretation of the output:

- The intercept is equal to 938.33. This is the average bone density for a person with normal density.
- The coefficient for “osteopenia” is equal to -138.33. This means that individual’s with osteopenia had bone densities that are on average 138.33 lower than individuals with normal bone density.
- The coefficient for “osteoporosis” is equal to -223.33. This means that individual’s with osteoporosis had bone densities that are on average 223.33 lower than individuals with normal bone density.

Running lm() on categorical x data

The coefficient estimates based on the model agree with what we calculate by hand:

```
calcium_data %>%
  group_by(type) %>%
  summarise(mean = mean(calcium_intake))
```

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

## # A tibble: 3 x 2
##   type      mean
##   <fct>    <dbl>
## 1 normal    938.
## 2 osteopenia 800
## 3 osteoporosis 715
```

- Note that 800 is 138.33 lower than 938.33
- Note that 715 is 223.33 lower than 938.33

Tests and p-values based on the linear model

```
library(broom)
calcium_lm <- lm(calcium_intake ~ type, data = calcium_data)

tidy(calcium_lm)

## # A tibble: 3 x 5
##   term          estimate std.error statistic    p.value
##   <chr>          <dbl>    <dbl>    <dbl>    <dbl>
## 1 (Intercept)      938.      95.4      9.83 0.0000000625
## 2 typeosteopenia  -138.     135.     -1.02 0.322
## 3 typeosteoporosis -223.     135.     -1.65 0.119
```

- We can also interpret these p-values. What is the null hypothesis?
- For the second row, the null hypothesis is that the regression coefficient for osteopenia is equal to 0. That is, the mean for patients with osteopenia is equal to the mean for those with normal bone density.

- The p-value is 0.32, so we do not reject the null hypothesis.

predict function to get 95% confidence intervals

We can use R code to calculate the 95% confidence intervals for the means:

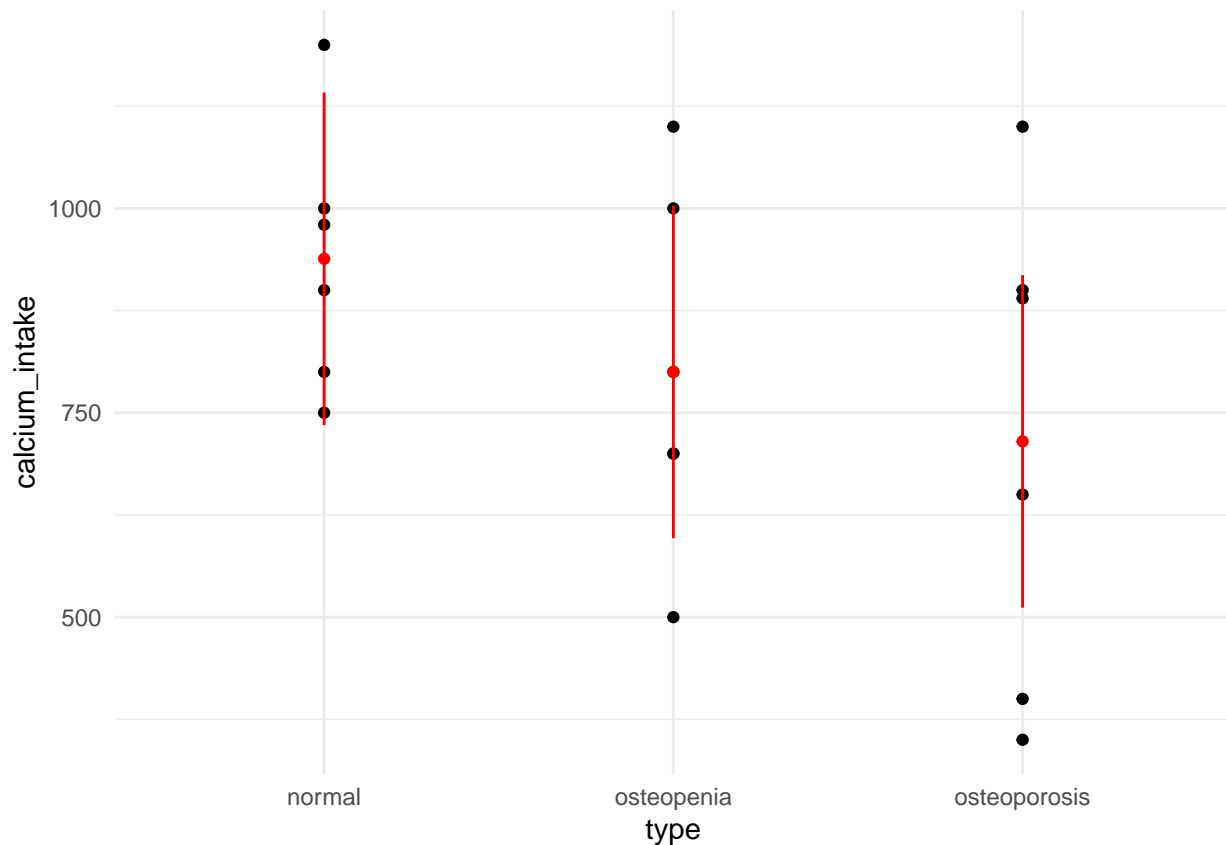
```
newdata = data.frame(type=c("normal", "osteopenia", "osteoporosis")) #make a tiny data frame storing the
predictions <- data.frame(predict(calcium_lm, newdata, interval="confidence")) #predict calcium intake
predictions$type <- c("normal", "osteopenia", "osteoporosis") #append another row to the data frame with
predictions
```

```
##      fit      lwr      upr      type
## 1 938.3333 734.9026 1141.7641    normal
## 2 800.0000 596.5693 1003.4307 osteopenia
## 3 715.0000 511.5693  918.4307 osteoporosis
```

Plot the observed data, the predicted means and their 95% CIs

What does this remind you of? It looks like the plots we made when we did ANOVA!

```
ggplot(data = calcium_data, aes(x = type, y = calcium_intake)) + geom_point() +
  geom_point(data = predictions, aes(y = fit), col = "red") +
  geom_segment(data = predictions, aes(y = lwr, yend = upr, xend = type), col = "red") +
  theme_minimal()
```



Run the model again using the type_reordered as the exposure variable

- Recall that type_reordered contains the same factor variable but with osteoporosis as the reference group

```
levels(calcium_data$type_reordered)
```

```
## [1] "osteoporosis" "normal" "osteopenia"
```

- Write the regression equation for the model with `type_reordered` as the `x` variable
- Before running the linear model, how do you expect the regression output to change?

Regression equation

$$\hat{y}_i = \hat{a} + \hat{b}_1 * [i's \text{ category} == \text{normal}] + \hat{b}_2 [i's \text{ category} == \text{osteopenia}]$$

Thus, a (the intercept) will be the average for patients with osteoporosis, $a + b_1$ will be the average for patients with normal bone density (implying the b_1 will be the additional calcium intake for patients with normal bone density) and $a + b_2$ will be the average for patients with osteopenia (implying that b_2 will be the additional calcium intake for patients with osteopenia)

The model

```
calcium_lm2 <- lm(calcium_intake ~ type_reordered, data = calcium_data)
```

```
tidy(calcium_lm2)
```

```
## # A tibble: 3 x 5
##   term                estimate std.error statistic    p.value
##   <chr>              <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)        715.       95.4       7.49 0.00000192
## 2 type_reorderednormal 223.      135.       1.65 0.119
## 3 type_reorderedosteopenia 85.      135.       0.630 0.538
```

- Can you interpret the intercept and other coefficients in this model?
- What is the average calcium intake for someone with osteopenia? Is it lower or higher than someone with normal bone density? By how much?

Another model: Mistaking categorical data for continuous data

- Recall that `type_num` is a numeric way of storing the categorical data stored in `type`

```
str(calcium_data$type_num)
```

```
## num [1:18] 1 1 1 1 1 1 2 2 2 2 ...
```

- What happens if we run the regression on `type_num`?

Another model: Mistaking categorical data for continuous data

Run the regression on `type_num`:

```
calcium_lm3 <- lm(calcium_intake ~ type_num, data = calcium_data)
```

```
tidy(calcium_lm3)
```

```
## # A tibble: 2 x 5
##   term                estimate std.error statistic    p.value
##   <chr>              <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)      1041.      141.       7.36 0.00000160
## 2 type_num         -112.      65.5      -1.71 0.107
```

- Notice that there is only one coefficient for `type_num`, but we were expecting two coefficients – one for the two non-referent levels of type.
- What happened?
- Answer: R interpreted `type_num` as a continuous, not categorical, variable. It estimated a regression slope term using $y = a + bx$, which is not what we want. This linear model makes the assumption that the increase in calcium intake going from category 1 to 2 and from 2 to 3 is the same. Thus, this model is not what we wanted to fit, and does not reflect the underlying data.
- Lesson: make sure that you double check how your categorical variables are coded! They should be stored as factors or else R will treat them as continuous variables.

Changing a variable type from continuous to categorical

- If your factor variable was coded numerically (like `type_num` in this example), R will code it as a continuous number and run simple linear regression on the underlying numbers. This is wrong, but can happen by mistake if you don't check.
- In this case you need to change the storage type using `your_data %>% mutate(var_categorical = as.factor(var_numeric))`
- In the code below, we update the storage type for `type_num` and store as a new categorical variable called `type_cat_ii`:

```
calcium_data <- calcium_data %>% mutate(type_cat_ii = as.factor(type_num))

str(calcium_data)
```

```
## 'data.frame': 18 obs. of 5 variables:
## $ calcium_intake: num 1200 1000 980 900 750 800 1000 1100 700 800 ...
## $ type : Factor w/ 3 levels "normal","osteopenia",...: 1 1 1 1 1 1 2 2 2 2 ...
## $ type_num : num 1 1 1 1 1 1 2 2 2 2 ...
## $ type_reordered: Factor w/ 3 levels "osteoporosis",...: 2 2 2 2 2 2 3 3 3 3 ...
## $ type_cat_ii : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 2 2 2 2 ...
```

Re-run the model on `type_cat_ii`:

```
calcium_lm4 <- lm(calcium_intake ~ type_cat_ii, data = calcium_data)

tidy(calcium_lm4)
```

```
## # A tibble: 3 x 5
##   term          estimate std.error statistic    p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)    938.        95.4      9.83 0.0000000625
## 2 type_cat_ii2  -138.       135.     -1.02 0.322
## 3 type_cat_ii3  -223.       135.     -1.65 0.119
```

- This looks better (we have two coefficients).
- But it is confusing, we have to know which numbers correspond to which categories. Better to code the underlying data using words (like the original `type` variable) to prevent mistakes and confusion.

Recap

- We now know how to write linear models when y is continuous and x is either continuous or categorical
- When x is categorical we expect $k - 1$ regression coefficients, where k is the number. The regression coefficients correspond to the average value of y for each category.
- When we have categorical data, it is important to make sure R knows it is categorical (by checking its `str()`) and setting the appropriate referent group.
- Hypothesis tests as how much each group differs from the referent group.

Future stats classes...

- We've only dealt with one x variable at a time. In future stat courses, you will learn how to model multiple x variables using multiple linear regression.
- This is important for prediction models (to get the best prediction you include every x variable that helps predict y)
- This is also important for causal models (for these models, you are interested in the causal effect of a specific x variable, but include other x variables to control for bias, such as confounding variables, and model interactions).