In-Class Lab 6

ECON 4223 (Prof. Tyler Ransom, U of Oklahoma)

February 8, 2022

The purpose of this in-class lab is to practice using dummy variables in R. The lab should be completed in your group. To get credit, upload your .R script to the appropriate place on Canvas.

For starters

Open up a new R script (named `ICL6_XYZ.R`, where `XYZ` are your initials) and add the usual "preamble" to the top:

```
# Add names of group members HERE
library(tidyverse)
library(broom)
library(wooldridge)
library(modelsummary)
```

Also install the package `magrittr` by typing **in the console**:

```
install.packages("magrittr", repos='http://cran.us.r-project.org')
```

and then add to the preamble of your script

```
library(magrittr)
```

The `magrittr` package contains extra features for writing even more expressive code.

Load the data

We'll use a new data set on extramarital affairs, called `affairs`.

```
df <- as_tibble(affairs)
```

Check out what's in the data by typing

```
datasummary_skim(df,histogram=FALSE)
```

You'll notice that there are a number of variables that only take on 0/1 values: `male`, `kids`, `affair`, `hapavg`, `vryrel`, etc. There are also variables that take on a few different values: `relig`, `occup`, and `ratemarr`.

Creating factor variables

Let's convert our 0/1 numeric variable `male` to a factor with levels `"yes"` and `"no"`:

```
df %<>% mutate(male = factor(male), male = fct_recode(male, yes = "1", no = "0"))
```

The `df %<>% mutate(...)` uses the `%<>%` operator which is shorthand for `df <- df %>% mutate(...)`. In other words, `%<>%` pipes forwards and then pipes everything backwards. This saves a little bit on syntax if you want to overwrite your data frame.

The first part of the `mutate()` function converts the 0/1 values to categories named `"0"` and `"1"`. The second part gives the categories more descriptive labels (`"yes"` and `"no"`).

Let's repeat this for some of the other variables: `ratemarr`, `relig`, `kids`, and `affair`:

```
df %<>% mutate(ratemarr = factor(ratemarr),
               ratemarr = fct_recode(ratemarr, very_happy = "5", happy = "4", average = "3",
                                     unhappy = "2", very_unhappy = "1")) %>%
        mutate(relig = factor(relig),
               relig = fct_recode(relig, very_relig = "5", relig = "4", average = "3",
                                  not_relig = "2", not_at_all_relig = "1")) %>%
        mutate(kids  = factor(kids), kids = fct_recode(kids, yes = "1", no = "0")) %>%
        mutate(affair = factor(affair), affair = fct_recode(affair, yes = "1", no = "0"))
```

where we used multiple pipe operators (`%>%`) to do each factor recode in a separate step. (We could have also put them all into one giant `mutate()` statement.)

Do another `datasummary_skim(df,histogram=FALSE)` to make sure the code worked in the way I told you it would. You'll notice that the factor variables now disappear from the `datasummary_skim()` output. To have them show up separately, follow the steps in the next section.

Summary stats of factor variables

You can look at the frequency of factor variables using the `datasummary_skim()` function from the `modelsummary` package, or by using the `table()` function:

```
datasummary_skim(df, type="categorical", histogram=FALSE)
table(df$relig)
table(df$ratemarr,df$kids)
```

You can also use the `prop.table()` function to get shares within-row (`margin=1`) or within-column (`margin=2`):

```
table(df$ratemarr) %>% prop.table()
table(df$ratemarr,df$kids) %>% prop.table(margin=1)
table(df$ratemarr,df$kids) %>% prop.table(margin=2)
```

You can also create a histogram of a factor variable in `ggplot()` as follows:

```
ggplot(df,aes(x=ratemarr)) + geom_bar()
```

This helps you visualize what share of the data falls into which category.

Multiple regression with factor variables

Let's run a regression with `naffairs` as the dependent variable and `male`, `yrsmarr`, `kids`, and `ratemarr` as the covariates.

```
est1 <- lm(naffairs ~ male + yrsmarr + kids + ratemarr, data=df)
```

Interpret the coefficient on `ratemarrvery_happy`.

Linear Probability Model

Let's run the same regression as before, but this time use `affair` as the dependent variable. What happens when you run the following code?

```
est2 <- lm(affair ~ male + yrsmarr + kids + ratemarr, data=df)
```

R doesn't want you to run a LPM because R was designed by statisticians who focus more on the "cons" of LPMs than on the "pros."

To run the LPM, adjust the code by using `as.numeric(affair)` as the dependent variable. Interpret the coefficients on `ratemarraverage` and `kidsyes`.

Interaction terms

Finally, let's run a more flexible model where we allow the effect of fathers and mothers to be different. The way to do this in `lm()` is as follows:

```
est3 <- lm(as.numeric(affair) ~ male*kids + yrsmarr + ratemarr, data=df)
print(tidy(est3))
```

The coefficient on the interaction term is labeled `maleyes:kidsyes`. Do fathers have a differential rate of extramarital affairs compared to mothers?