Exercise 12: Peer effects

Kristian Urup Olesen Larsen[1]

[1]kuol@econ.ku.dk

May 20, 2020

Problem 12.1.1. - Building the dataset (1)

Our first task will be to merge the dataset with itself to form pairs of roommates. First of though, let us load in the data

```
import numpy as np
import pandas as pd

df_single = pd.read_csv('peer_effects_room.csv')
```

To create the roommate pairs I merge the dataset with itself on the room column. This results in four rows per room because persons are also merged to themselves, but these additional rows can easily be filtered away:

```
df = df_single.merge(df_single[['person', 'room']],
                     on = 'room',
                     suffixes = ('', '_other')
                     )
df = df[df['person'] != df['person_other']]
```

Problem 12.1.2. - Building the dataset (2)

Now to add GPA data for the "other" roommate simply takes another merge

```
df = df.merge(df_single,
              left_on = 'person_other',
              right_on = 'person',
              suffixes = ('', '_other')
              )
```

After this step take a look at the resulting dataframe and validate that it looks as expected.

Problem 12.1.3. - Visualization (1)

Now on to visualizing the data. At this point I believe all of you have enough experience with matplotlib to solve this problem without help from me. Instead I want to use a few lines to talk about good visualization habits. These will cary over regardless of what program you are using to create your figures.

The most important thing to take away is that your figures should be *visually appealing*. If you do any kind of visual art you know very well that some aspects are inherently ugly while others are nice to look at. Some colors go well together. Others makes your eyes scream. You cannot combine ultra thin and ultra thick lines without causing some "aesthetic disturbance".

The brain works the same way whether it is looking at art or a graph - it relaxes when the figure is well balanced and becomes uneasy when it is not (try to find some bad graphs and see how you feel looking at them). Use this knowledge to create figures that communicate their content effectively without letting the visuals get in the way. In most cases you want your figures to fit into a place where they are so “aesthetically normal” that people don’t really notice them - figures should serve as a vehicle for information. If people do really notice your figure it is usually a bad sign, as they will spend time taking in the visuals instead of the information. This does not mean you should use the same template as everyone else. Templates become cliches and cliches are noticeable (think about figures made in excel). It also does not mean that your figures should be bland or boring. Boring is still boring. Aim for good looking, but in a very recognizable way, sort of like how pop music is good sounding but in a very recognizable way.

Aspect

The aspect of a figure is the width to height ratio. You typically set the aspect when defining a figure via the figsize argument. For instance this

```
fig, ax = plt.subplots(figsize = (4,3))
```

creates a figure with aspect 4:3. Aspect is very much related to the medium you are showing your figure in. For example try making a figure with 16:9 aspect. On your PC this will look uncomfortably wide because 16:9 is also the aspect of your screen. Try reducing the aspect to 13:9. A nice looking aspect is also related to the data you want to show. Some time series are very suited for ultra wide figures. Other types of data only work in 1:1 and so on.

Colors

Colors are of course central to any figure. There are essentially two types of colors: continous colormaps and discrete colormaps. Continuous colormaps contain smoothly transitioning colors which are good for coloring by a continous variable. The majority of continous colormaps are either sequential (e.g. light-to-dark/red-to-green etc) or divering (e.g. dark-to-light-to-dark/green-black-green). Of course your choice between these two types will depend on the kind of data you have. The sequential colormaps are useful for plotting things like temperature where higher values always should imply the same change in color. Divering colormaps have speciality uses, for instance when plotting data related to american politics it can be useful to have a blue-white-red colormap.

Deciding on a colormap continous colormaps is unfortinately not as easy as selecting the best looking one. There are two important things to consider when choosing a continous colormap

- Perceptual uniformity - Humans don’t perceive color distances very well. We pick up easily on small differences in color between red and yellow (i.e. in the orange colors) but are very bad at picking up differences between yellow and blue (i.e. in the green colors). For this reason plotting a continuous variable can often lead to problems where differences in specific ranges of the data can simply not be seen by humans. Perceptually uniform colormaps solve this problem by being specifically designed to match how humans perceive color differences. With a perceptually uniform colormaps the difference between 5 and 10 will look about as large as the difference between 45 and 50 to the human eye.

- Colorblindness - a significant proportion of your audience is colorblind. Catering to all of them can be difficult since there are different kinds of colorblindness with different consequences for color perception. You want to use a colormap that still communicates the same information when viewed by a colorblind person. It is impossible to maintain perceptual uniformity for all of your colorblind audience since many of them are simply missing the ability to pick up on parts of the spectrum, but your colormap should avoid dead-spots where a range of data will register as the exact same color.

Luckily there are excellent colormaps included in matplotlib that were designed to be perceptually uniform and colorblind-friendly. They are *viridis, magma, inferno and plasma*. Use them.

Discrete colormaps consist of a number of easily separable colors that are suited for plotting categorical data. Sometimes these are o.k. to use, but often you will find that you have a personal preference for which colors you use. In this case you can specify them manually. You should still be aware of your colorblind audience - dont pick red and green as your primary colors.

Weights and styles

Weight and style is a broad range of aesthetic options covering the width of lines, the size of scatterplot markers, dashed and dotted lines and so on. The possibilities here are endless so I will not spend time discussing each and every one. You should however know a few things:

- Line widths and marker sizes needs to be balanced in a figure. To large differences throws off your audience and generally looks bad. Use your aesthetic intuition.
- Dashed lines and hollow/different shaped markes can be a good alternative to colors. They generally do better in black and white and dont run the risk of making your figures unintelligible to any colorblind readers. You do however need to be wary of overusing them. If your figure becomes to complicated to look at people spend more time digesting the visual impression than the information.

Problem 12.1.4. - Visualization (2)

Once again I hope that you can make the figure yourself. What you should see is that there appears to be no relation between the GPA of either roommate. This is good - our research design will hinge on the fact that people are randomly assigned to rooms. (Why is randomization so important? Try to articulate exactly what it is about randomization that makes it a crucial ingredient of any good research design).

We can also do the same check in a regression. This is of course useful to know because it allows us to check for conditional independence. We could imagine that rooms were randomly assigned conditional on gender, which would be impossible to capture in the visual inspection, but can easily be checked in a regression.

```python
from statsmodels.formula.api import ols

model = ols('high_school_GPA ~ high_school_GPA_other', data = df).fit()
model.summary()
```

When you run this regression you should see that the confidence interval on high_school_GPA_other contains 0. Of course this it not a proof of randomness, in the real world it could be a result of attenuation bias, omitted variable bias etc.

Problem 12.1.5. - Exogeneous peer effects

In this problem we want to look at *exogeneous* peer effects, i.e. effects that improves peer performance but are determined outside of (in this case prior to) the peer group. Because peers are randomized into groups this is as simple as regressing

```python
model = ols('college_GPA ~ high_school_GPA_other', data = df).fit()
model.summary()
```

The interpretation of the coefficient on high_school_GPA_other is then the causal effect on a students GPA of living with a roommate with a 1 point increase in GPA from high school.

We do not need to control for the primary individuals high school GPA exactly because roommates are assigned randomly, so the high school GPA of the peer will be orthogonal to the high school GPA of the primary person. We also do not control for the other persons college GPA because we want the total effect regardless of whether it works directly or through the peers college performance. (Drawing a causal diagram might help you follow this discussion).

Problem 12.1.6. - Endogeneous peer effects

Next the endogeneous effects. These are effects that arise within the peer group independent of the group members prior skills. The relevant variable here is the peers college GPA but importantly conditional on the peers high school GPA. We want to know if living with someone who outperforms their expected GPA (given by their high school GPA) has a causal effect on ones own GPA.

```
model = ols('college_GPA ~ college_GPA_other + high_school_GPA_other', data = df).fit()
model.summary()
```

Again we don't need to include high_school_GPA (but can gain some precision by doing so).