GBA 464: Baltimore

Yufeng Huang

September 26, 2021

1 Overview

This document describes the third take-home (ungraded) exercise of GBA 464, involving the Baltimore public sector employment data. We start with basic merge and cleaning exercises for the employment/salary data, from Baltimore public sector. The point is to further practice merge and data cleaning, and also manipulations on character data. After data cleaning, we will use this data set to evaluate wage growth and dispersion (i.e. how different are wages across people) in the Baltimore public sector. Finally, we will borrow an external package that searches names in gender database and matches names to gender. We'll use this result to evaluate whether there are gender gaps and racial gaps in wages and how severe this is.

2 Step 1: data cleaning

There are 11 Dropbox links, which are 11 years of Baltimore City public sector employment data. These are individual name, department they work in, positions they hold and their annual salary. By some legal requirements these data should be public, and we obtain them from Open Data Baltimore.

Note that the way we read data is very cumbersome and prone to mistakes. You can construct a loop to read the data just like Example 2 in flow control lecture 2.

3 Step 2: construct variables and summary statistics

A large part of the data set is in strings. We first need to convert the strings (i.e. character data) into numeric. Do it for $annualsalary and $grosspay (actually we don't need grosspay later, but why not do it as well). Note that as.numeric will not work.

After this is done, calculate a few important variables. We want: 1) duration of employment in the sample; 2) year in which this person was hired; 3) average annual salary within the duration of employment; and 4) average growth rate of annual salary within the duration of employment.

Optionally, you can also compute some other interesting numbers as well. Don't be constrained by me!

4 Step 3: organize names

We next want to find out whether key statistics that we computed above are different between female and male, and between non-caucasian and caucasian. This is a sensitive topic but I think it is important for us to find out ourselves (rather than listening to politicians). All we know are names, positions (departments) and salaries, and we don't know gender and race. But there has been many studies showing that names are highly correlated with gender and race; and we can use this relationship to dig out the information we're after.

First, the thing you can do it yourselves is to split names. By now we have covered splitting a string and regular expressions. Let's split the full name (and note that names can be rather long) into first and last name, and discard any middle names.[1] Before we cover repetition structures, we have talked about how to do this with a vector of three names. Now, we can do this in the "for-loop" way. Try to write a repetition structure that splits names in to first name and last name, stored in separate columns.

If your code is too slow, you can stop the code (or loop over fewer rows just to see the results

[1] Some foreign names might not be perfectly recorded; e.g. Jean-Francois might be Jean Francois. For a quick-and-dirty first cut just treat Jean as the first name, in cases like this.

work). We will need to talk about how to improve this in the following weeks!

5 Step 4: use the library 'gender' to predict gender and country of origin

When the above is done, I propose we install the 'gender' library and use it to figure out the likely gender and indirectly also figure out whether a person is likely caucasian. However, if your folder for R libraries contain spaces or non-standard characters (e.g. non-English or some symbols) then the installation might be difficult. In this case try changing the library folder.

When using 'gender' library, notice that we can specify which database to use when we match names to gender. There are some US database (specified by 'method' and 'country' arguments) and some European country database. If you try matching the same set of names to different databases you'll find out that a smaller set of names are matched to a European database, compared to a US database. In addition, eyeballing the discrepancies and you'll find that a lot of the names that are in the US data but not European data are likely non-caucasian (representing that there's larger diversity in the US population). We can use this discrepancy to define likely caucasian versus likely non-caucasian.

Finally, after categorizing the sample into likely gender and race, we can see whether salary, employment duration, growth rate and other statistics changes with race and gender. We'll work this out in class on Thursday. A step forward is to look at whether the position one holds changes with race and gender, and whether the position itself explains all the discrepancies in salary etc. (In other words, are female and minorities discriminated by the glass ceiling in rank or by salary itself given the same rank?) This question might come in the weekend assignment but it depends on our progress in class.

6 Step 5: the conclusion

Do female/modern names correspond to lower payment or any other disadvantages, reflected in job duration, salary growth, etc? We can directly aggregate the data by group (e.g. combinations of female/modern) to see this. Alternatively, we can also define a function that queries the average of a variable conditional on the criteria we give set. For example, if I want average annual salary and salary growth for females with modern names, my function should organize the results in a vector/list. If I want average job duration for males with classical family names, we can put different arguments in the same function to do it. In this simple example, there's actually no clear reason why we want to write a function; but you will find this useful when you deal with more complicated data sets.