Background

In this problem set you will estimate a key policy parameter: the interest rate elasticity of mortgage demand, which tells us how much mortgage demand changes for a given change in mortgage rates. This is important for a number of reasons.First, it determines the degree to which monetary policy affects aggregate borrowing behavior, since mortgages account for most of household debt. Second, it helps policymakers understand the effect of the home mortgage interest deduction (a 80 billion dollar program).

The estimation of this elasticity has been met with limited success due to data limitations and a lack of exogenous variation in interest rates. However, Defusco and Paciorek (2017) introduced a novel method to estimate the interest elasticity of mortgage demand. In particular, they exploited a regulatory requirement imposed on the Government Sponsored Enterprises (GSEs) that generated exogenous variation in the relationship between loan size and interest rates: loans above the conforming loan limit (CLL) typically have higher interest rates than those below. In Figure 1A, they plot the mean interest rate of loans by loan size, and interest rates are clearly discontinuous around the CLL.

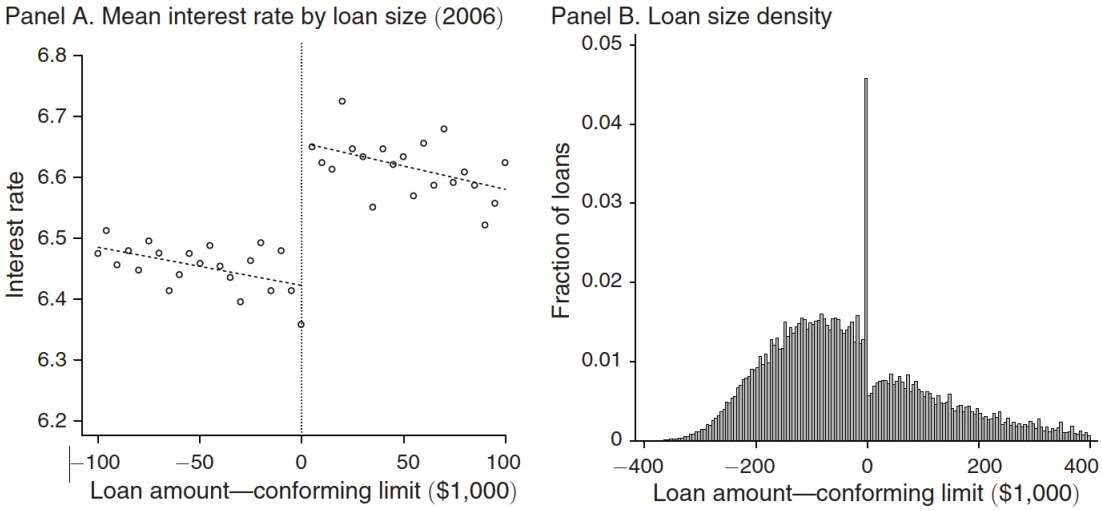

FIGURE 1

Intuitively, this suggests that some borrowers who would have taken out loans above the CLL are induced to take out loans at the CLL instead. Indeed, they find evidence of this, where there is significant bunching of loans right at the CLL, as shown in Figure 1B. In this assignment, you will exploit this exogeneous variation to estimate the interest rate elasticity of mortgage demand.

Question 1: Pre-processing Data

As this section serves to prepare the data for data analysis, it is advisable to read Question 2 prior to attempting Question 1.

1. **Data types:** Load the main dataset `all_llma_nc.csv` into your IDE (it might take a while), and check the data types for each variable. Report any changes made to the data types throughout the assignment here.

2. **Subsetting data:** For this assignment, we are interested in 30 year fixed mortgages that originated between the years 2000 and 2007. In the data, these are loans with a `product_type` of 10, `original_term` of 360 or 180, `loan_type` of 1, and `loan_purpose` of 1, 2, 3, or 5. Drop the unnecessary observations.

3. **Combining data from multiple datasets:** The file `conformingloanlimits.dta` contains the conforming loan limits for each year. Match each loan in the main dataset to its respective conforming loan limit. *Hint: merge the two datasets.*

4. **Calculating deviations:** Normalize the loan deviations as per **paper**: center each loan at the (log) conforming limit in the year the loan was originated, such that a value of zero represents a loan size exactly equal to the conforming limit, while anything else represents (approximate) percentage deviations from the conforming limit. In other words, create a new variable `deviation`, which takes the log of a function of both the loan and the loan limit.

5. **Grouping loans into bins:** Group the deviations into bins with indices centered at 0. In general, you may find yourself changing the number of bins as you go along; do not to hard code the number of bins. *Hint: create a separate dataset for the bins.*

6. **Calculate the share of loans per bin:** For each bin, (1) compute the share of loans and store this as a new variable `share`, and (2) compute the mean initial interest rates for the across the loans.

7. **Interest variation:** What is the initial interest rate for loans just below the conforming loan limit? What is the initial interest rate for loans just above the limit?

Question 2: Data analysis and Visualization

1. **Recreating Figure 1A:** For loans originating in the year 2003 that fall within \$100,000 of the CLL, plot the mean interest rate (at the bin level) as a function of the loan amount relative to the conforming limit. Include both the scatter plots and the best fit lines.

2. **Recreating Figure 1B:** Plot the share of all loans that are in each bin against the bin indices. This sample includes loans that fall within \$100,000 of the CLL.

3. **Estimate bunching:** Bunching is estimated as the difference between the observed and counterfactual bin counts in the excluded region at and to the left of the conforming limit. To estimate bunching at the conforming limit, you should do the steps listed below, referring to the paper for specifics. The parameters chosen by the authors are in brackets for your reference. Run the bunching algorithm and report the estimated normalized excess mass and elasticity, with the corresponding graph attached.

   (a) Define an exclusion region around the conforming limit (lower limit of 0.025).

   (b) Choose a degree of polynomial (thirteenth-degree polynomial).

   (c) Choose the width of bins (1-percent).

   Hint: you should be able to estimate the bunching easily with the any of the various bunching packages available in both STATA and R. I use the `bunchit` *function from the* `bunching` *library to illustrate this. The function, as specified in the documentation, is as follows:*

```
bunchit(z_vector, binv = median, zstar, binwidth, bins_l, bins_r,poly = 9,
    bins_excl_l = 0, bins_excl_r = 0, extra_fe = NA,rn = NA, n_boot = 100,
    correct = TRUE, correct_above_zu = FALSE,correct_iter_max = 200, t0, t1,
    notch = FALSE, force_notch = FALSE,e_parametric = FALSE, e_parametric_lb
    = 1e-04, e_parametric_ub = 3,seed = NA, p_title = , p_xtitle = deparse(
    substitute(z_vector)),p_ytitle = Count, p_title_size = 11,
    p_axis_title_size = 10,p_axis_val_size = 8.5, p_miny = 0, p_maxy = NA,
    p_ybreaks = NA,p_freq_color = black, p_cf_color = maroon,p_zstar_color =
    red, p_grid_major_y_color = lightgrey,p_freq_size = 0.5, p_freq_msize =
    1, p_cf_size = 0.5,p_zstar_size = 0.5, p_b = FALSE, p_e = FALSE,
    p_b_e_xpos = NA,p_b_e_ypos = NA, p_b_e_size = 3, p_domregion_color =
    blue,p_domregion_ltype = longdash)
```

   Every function has the same underlying structure. The function is `bunchit(...)`, and you call it by replacing the stuff in the brackets `(...)` with your parameters of interest.

   The parameters in this function include: `z_vector,binv,zstar,` and so on. If the parameter is followed by an equal sign, the value that follows is the parameter's default. In other words, if you choose not to supply the function with a value for said parameter, the default is assumed. Otherwise, the parameter will take on the supplied value. Similarly, you must supply values for parameters with no default values for the function to run.

   The full documentation is available here. As stated in the documentation, `z_vector` is a numeric vector of (unbinned) data. Therefore, although you have previously binned the data, you have to use the original data to run this function. `zstar` is the bunching point, which is the CLL. However, this function is defined such that setting `zstar=0` would throw an error, so you can either re-center the data or simply redefine the

function. `bins_l` and `bins_r` are the number of bins to the left and right of zstar respectively. `poly` is the order of polynomial for the counterfactual fit, with a default of 9. You can change the value to test different polynomials degrees. An example of the function call is as follows:

```
bunchit(runif(1000, 20, 40), zstar = 30, t0 = 0.03, t1 = 0.09, binwidth =
    20, bins_l = 50, bins_r = 50, poly = 13)
```

4. **Calculating elasticity:** The semi-elasticity is defined as $\epsilon_s^r = \frac{\Delta \hat{m}}{\Delta R * \hat{R}}$, where $\Delta \hat{m}$ is the approximate percentage change in mortgage demand induced by the conforming limit (what is reported by `nearbunch`: it is the difference between the actual and counterfactual distribution at the bunch point), while the denominator is the change in interest rates. Using the values estimated with the bunching function, calculate the elasticity of mortgage demand. The interpretation of this semi-elasticity is that a hundred basis point increase in mortgage rates (say from 4% to 5%) leads a ϵ_s^r% decline in mortgage demand. Table 3 shows the estimates from Defusco and Paciorek (2017) of this semi-elasticity from CA. Do you find similar numbers using info from NC?

R Hints

R command	Description
`getwd()`	Returns working directory
`setwd("Dropbox/bunching")`	Sets working directory
`read.csv(csvFileName)`	Reads csv files
`read.dta13(dtaFileName)`	Reads dta files
`str(data)`	Returns the structure of the data
`substr(char)`	Substrings a character

`inner_join(x,y,...)`
`full_join(x,y,...)`
`left_join(x,y,...)`
`right_join(x,y,...)`
	Merges two dataframes, returning only the merged columns

`filter(condition)`	Returns observations that satisfy the condition
`mutate(.,newCol=0)` `df['newCol']=0`	Creates a new column in the dataframe (df) with initial values 0

A primer to data manipulation in R

This section addresses some general tips for dealing with data, and should not be taken as an exhaustive guide to R - you should probably refer to the R book (**rBook**) for that.

Data types and data structures

While there are several data types in R, the ones you'd have to deal with most commonly are **characters**, **numerics**, **integers**, **logicals**, and **factors**. While most of these are fairly straightforward, converting from a **factor** (i.e., the levels are really just integers) to an **integer** requires the intermediate step of conversion to a **character**. This is because the **level** may not correspond to the **integer** in a vector. The following example is an illustration:

```
> x = c("1","2","1", "1", "10")
> as.factor(x)
[1] 1  2  1  1  10
Levels: 1 10 2
> myLevels = as.factor(x)
#see that this doesn't return the actual integers
> as.integer(myLevels)
[1] 1 3 1 1 2
#so you should do this instead
> as.integer(as.character(myLevels))
[1] 1  2  1  1  10
```

Depending on the data, you would store it with different data structures. Two things to consider are the dimension of your data and, as alluded to above, the data type. Some data structures, like the atomic vector and matrix, can only be used to store data of the same type. Others, like lists and dataframes (which are probably the most commonly used for our purposes), can be used to store different data types.

Useful packages

`tidyverse` is probably the best thing that has happened to R, and `dplyr` is especially useful for data manipulation. You should familiarize yourself with pipe operators, `%>%`, which makes writing readable code really easy!

While `.csv` files are definitely the way to go, you'll probably have to deal with several `.dta` files because some people continue to use STATA. Depending on how old the file is, you would require different packages (and functions) to read them. If the `.dta` file was created from STATA12 and before, you would use `read.dta(filename)`, from the `foreign` library. For STATA15 and before, you could re-save the data file as an older version in STATA, or use `read.dta13(filename)`, from the `readstata13` library. As you can see, this changes ever so often, so just do a google search for whatever you need.

Lastly, when installing multiple packages, store the packages you want to install in a vector and run `install.packages(c("package1","package2"))` instead of calling the `install.packages()` function multiple times for each package.